University of Alberta

Rayset and Its Applications to Static and Dynamic Image Synthesis

by

Minglun Gong

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-87977-1
Our file Notre référence
ISBN: 0-612-87977-1

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

Canada

University of Alberta

Library Release Form

Name of Author: Minglun Gong

Title of Thesis: Rayset and Its Applications to Static and Dynamic Image Synthesis

Degree: Doctor of Philosophy

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Date: Aug 25, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Rayset and Its Applications to Static and Dynamic Image Synthesis** submitted by **Minglun Gong** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Yee-Hong Yang

Dr. Pierre Boulanger

Dr. Sherif Ghali

Dr. Arturo Sanchez-Azofeifa

Dr. Wolfgang Heidrich

Supervisor writes the date that the thesis is approved by committee here Aug. 22, 2003

To my mom and dad
for their moral and financial support
over the many, many
years it took for this degree

Abstract

Image-based rendering, IBR for short, is an area that gained in popularity during the last decade. There are two main objectives in this research: to provide a framework for IBR techniques and to apply IBR to dynamic scenes.

A novel taxonomy, called *the rayset*, is presented in this dissertation. The new taxonomy studies the scene representation techniques and the scene reconstruction techniques used in IBR separately. It is demonstrated that different image-based scene representations can all be cast as different kinds of raysets, and different scene reconstruction approaches can be regarded as attempts to render different raysets. Based on the taxonomy, existing IBR approaches are classified and the relationship between them is identified.

The rayset taxonomy also facilitates the development of new IBR techniques. Three novel IBR approaches, which are applications of the rayset concept in static environments, are presented. Each approach has different requirements and different advantages and can therefore be used for different applications.

Dynamic IBR research, which is still in its infancy, studies how to generate novel videos based on captured videos. One critical requirement for dynamic IBR is to reduce the sampling density. Based on the rayset concept, a new technique that can handle sparsely sampled scenes is discussed. A novel stereo vision algorithm is also presented, which provides accurate disparity information for the rendering process.

In summary, this dissertation provides several original contributions in the IBR area. Possible future research directions on how to employ temporal coherence in dynamic scenes are discussed.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to those that have supported my research.

First of all, great thanks go to my supervisor, Dr. Herbert Yang, for taking me on as his student and allowing me to pursue my research interests. He is, without question, the most enjoyable person I could imagine having as a research advisor. Without his guidance, encouragement, and valuable suggestions, the dissertation presented here would not have been possible.

I am very grateful to the members of my examining committee: Dr. Pierre Boulanger, Dr. Sherif Ghali, Dr. Arturo Sanchez-Azofeifa, and Dr. Wolfgang Heidrich for taking the time to read and comment on this work.

I would also like to thank current and former members in Computer Graphics group, at the University of Saskatchewan and at the University of Alberta, in particular Byron Bashforth, Yufei Zhang, Jianwei Song, Xuejie Qin, Hai Mao, Daniel Neilson, Yi Xu, and Jiayuan Zhu, for many illuminating discussions.

In addition, I want to acknowledge Dr. Marc Levoy of Stanford University, Dr. Yuichi Ohta of the University of Tsukuba, Dr. Daniel Scharstein of Middlebury College, Mr. Larry Zitnick of Carnegie Mellon University, and Mr. Daniel Wood of University of Washington for providing datasets used in this dissertation.

Financial supports from, NSERC, the Killam Trust, the Department of Computing Science at the University of Alberta, and the Department of Computer Science at the University of Saskatchewan are gratefully acknowledged.

Table of Content

Chapter 1 Introduction **1**

1.1 CONSIDERATION ON MODELING AND RENDERING 2

1.1.1 Models *2*

1.1.2 Images *3*

1.1.3 Modeling *4*

1.1.4 Rendering *5*

1.2 MOTIVATION 6

1.2.1 Taxonomy for Image-based Rendering *6*

1.2.2 Dynamic Image-based Rendering *7*

1.3 ORGANIZATION OF THE DISSERTATION 8

Chapter 2 Previous Work **10**

2.1 IMAGE-BASED MODELING 10

2.1.1 Models from Silhouettes *11*

2.1.2 Models from Color Matching *12*

2.1.3 Models from Range Images *14*

2.1.4 Models from Human Interaction *15*

2.2 IMAGE-BASED RENDERING 16

2.2.1 Images as Textures *16*

2.2.2 Images as References *18*

2.2.3 Images as Layers *21*

2.2.4 Images as Primitives *23*

2.2.5 Images as Datasets *25*

2.3 STEREO VISION 28

2.3.1 Local Methods *28*

2.3.2 *Cooperative Methods* 29
2.3.3 *Optimization Methods* 30
2.3.4 *Multi-view Methods* 32
2.4 IMAGE MORPHING 33
2.4.1 *Feature Specification* 34
2.4.2 *Warping Generation* 34
2.4.3 *Transition Control* 35
2.4.4 *Other Related Work* 36
2.5 IMAGE CACHING 36
2.5.1 *Pre-rendering* 37
2.5.2 *Dynamic Updating* 38
2.5.3 *Sampling plus Shading* 40
Chapter 3 Taxonomy for Image-based Rendering 41
3.1 RAYSET TAXONOMY 42
3.1.1 *Definition of Rayset* 42
3.1.2 *Classification of Scene Representations* 43
3.1.3 *Classification of Scene Reconstruction Techniques* 44
3.2 RAYSET EXAMPLES 45
3.2.1 *Planar Images* 45
3.2.2 *Non-planar Images* 49
3.2.3 *High-dimensional Representations* 50
3.3 RAYSET RENDERING 53
3.3.1 *Plenoptic Sampling* 53
3.3.2 *Depth Warping* 54
3.3.3 *Texture Mapping* 55
3.3.4 *Scene Morphing* 56
3.3.5 *Comparison among Categories* 57
3.4 RAYSET EDITING 58
3.4.1 *Distortion Filters* 58
3.4.2 *Re-shading Filters* 61

Chapter 4 Rayset Applications for Static Scenes 62

4.1 OBJECT CENTERED CONCENTRIC MOSAICS 63

4.1.1 Parameterization Scheme 63

4.1.2 Creation of Object Centered Concentric Mosaics 64

4.1.3 Rendering of Object Centered Concentric Mosaics 66

4.1.4 Experimental Results 68

4.1.5 Discussions 69

4.2 SPIRAL TEXTURE MAPPING 70

4.2.1 Uniform Sampling for the Viewing Directions 71

4.2.2 Adaptive Sampling for the Projection Positions 73

4.2.3 Comparison with the Two-plane Parameterization 74

4.2.4 Creation of Spiral Textures 76

4.2.5 Rendering of Spiral Textures 78

4.2.6 Experimental Results 81

4.2.7 Results Evaluation 84

4.2.8 Discussions 87

4.3 LAYER-BASED MORPHING 88

4.3.1 Warping Based on Regions 88

4.3.2 Construction of Multi-layered Images 89

4.3.3 Morphing between Multi-layered Images 91

4.3.4 Experimental Results 92

4.3.5 Comparison with Feature-based Morphing 93

4.3.6 Discussions 95

Chapter 5 Reliability-based Stereo Matching 96

5.1 CONSISTENCY OF DISPARITY MAPS 97

5.1.1 The Stereo Vision Problem 97

5.1.2 Definitions of Constraints 98

5.1.3 Comparison with Other Constraints 100

5.2 RELIABILITY THRESHOLDING PROCESS 101

5.2.1 Efficiency Improvement 102

5.2.2 *Reliability Calculation* ... 103
5.2.3 *Ground Control Points* ... 105
5.3 ENFORCING CONSISTENCY THROUGH RELAXATION ... 105
5.3.1 *Unambiguous Stereo Matching* ... 106
5.3.2 *Multi-view Stereo Matching* ... 107
5.4 EXPERIMENTAL RESULTS ... 108
5.4.1 *Reliability Thresholding* ... 108
5.4.2 *Unambiguous Stereo Matching* ... 109
5.4.3 *Multi-view Stereo Matching* ... 112

Chapter 6 Rayset Application for Dynamic Scenes ... 114

6.1 CAMERA FIELD PARAMETERIZATION ... 115
6.1.1 *General Parameterization Scheme* ... 115
6.1.2 *Special Cases* ... 116
6.2 COLOR-MATCHING INTERPOLATION ... 117
6.2.1 *The Interpolation Problem* ... 117
6.2.2 *Matching along Epipolar Lines* ... 118
6.2.3 *Result of Color-matching* ... 121
6.3 DISPARITY-SEARCHING INTERPOLATION ... 122
6.3.1 *Searching Based on Disparities* ... 123
6.3.2 *Result of Disparity-searching* ... 127
6.3.3 *Computational Cost Analysis* ... 128
6.4 EXPERIMENTAL RESULTS ON STATIC SCENES ... 129
6.4.1 *Planar Camera Field Interpolation* ... 129
6.4.2 *Cylindrical Camera Field Interpolation* ... 130
6.4.3 *Comparison with Dynamic Reparameterized Rendering* ... 131
6.4.4 *Robustness Evaluation* ... 133
6.5 EXPERIMENTAL RESULTS ON A DYNAMIC SCENE ... 133
6.5.1 *Camera Calibration* ... 134
6.5.2 *Video Synchronization* ... 135
6.5.3 *Image Rectification* ... 136
6.5.4 *Disparity Pre-calculation* ... 138

6.5.5 Rendering Results *140*

Chapter 7 Conclusion and Future Work **142**

7.1 CONTRIBUTIONS 142

7.1.1 Rayset Taxonomy *142*

7.1.2 Novel Stereo Vision Algorithm *143*

7.1.3 Dynamic Scene Rendering Technique *144*

7.2 FUTURE WORK 145

7.2.1 Stereo-Motion Analysis *145*

7.2.2 Unstructured Camera Rendering *146*

References **147**

Appendix A: Reliability and Approximated Reliability **166**

Appendix B: Estimated Parameters for Camcorders **168**

List of Tables

Table 3-1: Comparison among different IBR categories 57
Table 5-1: Comparison for unambiguous stereo matching 110
Table 5-2: Comparison for multi-view stereo matching 112

List of Figures

Figure 1-1: Domains of the traditional computer graphics and computer vision. 1
Figure 1-2: The same object under different representations. 3
Figure 1-3: Planar pinhole camera model. 4
Figure 3-1: Classification of raysets. 43
Figure 3-2: Classification of 2D continuous raysets. 43
Figure 3-3: Classification of scene reconstruction techniques. 44
Figure 3-4: More detailed classification of scene reconstruction techniques. 45
Figure 3-5: Planar images (a) perspective image (b) parallel image (c) object image. 46
Figure 3-6: Representations based on perspective images. 47
Figure 3-7: Representations based on parallel images. 47
Figure 3-8: Representations based on object images. 49
Figure 3-9: Strip camera model. 50
Figure 3-10: Two-plane parameterization used in light field rendering. 51
Figure 3-11: Parameterization scheme used in concentric mosaics. 52
Figure 3-12: Effects of distortion filters on the "Buddha" light field. 59
Figure 4-1: Parameterization scheme used in object centered concentric mosaics. 64
Figure 4-2: Two object centered concentric mosaics for the "Happy Buddha" model. ... 65
Figure 4-3: Three steps in the rendering process. 67
Figure 4-4: Rendering results for different viewing directions. 68
Figure 4-5: Rendering results for different camera settings. 69
Figure 4-6: Rendering results for viewpoints with different heights. 69
Figure 4-7: Uniform sampling using spiral. 71
Figure 4-8: A ray under (a) the Cartesian space and (b) the line space. 73
Figure 4-9: Calculation of the view-independent component. 76
Figure 4-10: Calculation of the view-dependent component. 77

Figure 4-11: The closest directions sampled. 80
Figure 4-12: View-independent component for the "Fish" dataset. 81
Figure 4-13: Samples of the view-dependent component (after color compression). 82
Figure 4-14: Rendering process (a)(b) view-independent (c)(d) view-dependent..... 83
Figure 4-15: Rendering results evaluation for different views. 84
Figure 4-16: Rendering results using the low sampling rate data..... 85
Figure 4-17: Rendering results under different compression approaches. 85
Figure 4-18: Results rendered by the surface light field approach. 86
Figure 4-19: Layer-based morphing algorithm..... 91
Figure 4-20: Novel view generated by morphing for an architecture scene..... 92
Figure 4-21: Control features defined for feature-based morphing. 93
Figure 4-22: Feature-based morphing results. 93
Figure 4-23: Control features defined for layer-based morphing. 94
Figure 4-24: Layer-based morphing results. 94
Figure 5-1: Effects of using different constraints. 100
Figure 5-2: Algorithm for efficient cost calculation. 102
Figure 5-3: The best path and the alternate paths. 103
Figure 5-4: Algorithm for tracing the best and alternate paths. 104
Figure 5-5: Algorithm for handling ground control points. 105
Figure 5-6: Results of the reliability calculation and thresholding. 108
Figure 5-7: Unambiguous stereo matching results. 111
Figure 5-8: Trinocular and multi-view stereo matching results..... 112
Figure 6-1: Interpolation between reference images for planar camera field. 117
Figure 6-2: Interpolation between reference images for cylindrical camera field. 120
Figure 6-3: Algorithm for color-matching interpolation. 121
Figure 6-4: Interpolation results (a) linear (b) color-matching. 121
Figure 6-5: Incorrect interpolations using color-matching interpolation. 122
Figure 6-6: Intersection searching using disparity for planar camera field. 123
Figure 6-7: Intersection searching using disparity for cylindrical camera field. 124
Figure 6-8: "Rubber sheets" problem caused by linear interpolation. 126
Figure 6-9: Algorithm for disparity-searching interpolation. 127

Figure 6-10: Interpolation results (a) disparity-searching (b) combined. 127
Figure 6-11: Interpolation results for the "Santa Claus" planar camera field..... 129
Figure 6-12: Interpolation results for a cylindrical camera field. 131
Figure 6-13: Comparison with dynamically reparameterized light field rendering. 132
Figure 6-14: Rendering results using relative noisy disparity maps..... 133
Figure 6-15: Experiment setup for dynamic IBR. 134
Figure 6-16: Radial distortion correction..... 135
Figure 6-17: Synchronized frames in captured videos. 136
Figure 6-18: Image rectification using vanishing points. 137
Figure 6-19: Different views after rectification. 138
Figure 6-20: Disparity maps calculated using different algorithms..... 139
Figure 6-21: Disparity maps calculated for different views. 140
Figure 6-22: Interpolation results using different approaches. 141
Figure 6-23: Different frames in the generated video..... 141

List of Equations

Equation 1-1: Sampling of the image parameter space. 3
Equation 3-1: Original plenoptic function. 42
Equation 3-2: 5D plenoptic model. 42
Equation 3-3: Definition of rayset. 42
Equation 3-4: Support function for perspective images. 46
Equation 3-5: Functions for converting vectors to angles. 46
Equation 3-6: Support function for parallel images. 47
Equation 3-7: Support function for object images. 48
Equation 3-8: Support function for cylindrical panoramas. 49
Equation 3-9: Support function for images taken by a strip camera. 50
Equation 3-10: Support function for light slabs. 51
Equation 3-11: Support function for concentric mosaics. 52
Equation 3-12: Effect of a warping on an image. 59
Equation 3-13: Random distortion filter for light slabs. 60
Equation 3-14: Wide object angle filter for light slabs. 60
Equation 3-15: Horizontal ripple filter for light slabs. 60
Equation 3-16: Vertical ripple filter for light slabs. 60
Equation 3-17: Circular ripple filter for light slabs. 61
Equation 4-1: Support function for object centered concentric mosaics. 64
Equation 4-2: Quantization of depth information. 65
Equation 4-3: Reverse mapping equations for object centered concentric mosaics. 67
Equation 4-4: Parametric functions of a spiral defined on a sphere. 72
Equation 4-5: Length of the spiral that covers the hemisphere. 72
Equation 4-6: Elliptic integral of the second kind. 72
Equation 4-7: Total number of sampling directions needed. 72

Equation 4-8: Calculation of the spiral parameter for a given sample. 72
Equation 4-9: Scaling factor for adaptively sampling. 74
Equation 4-10: Support function for spiral textures. 74
Equation 4-11: Number of samples needed for spiral texture parameterization. 75
Equation 4-12: Number of samples needed for two-plane parameterization. 75
Equation 4-13: Ratio of the numbers of samples needed. 75
Equation 4-14: The number of rounds for spiral curve..... 80
Equation 4-15: Constraint on the warping function used. 89
Equation 4-16: Support area of a layer. 90
Equation 4-17: Binary image generated based on the support area..... 90
Equation 4-18: Support image used by the hierarchical polynomial fit filter..... 91
Equation 4-19: Interpolation of the corresponding pixels in layer images..... 91
Equation 5-1: Strong consistency constraint..... 98
Equation 5-2: Weak consistency constraint..... 98
Equation 6-1: Support function for camera fields..... 115
Equation 6-2: Support function for planar camera fields..... 116
Equation 6-3: Support function for cylindrical camera fields. 116
Equation 6-4: Interpolation on camera plane..... 117
Equation 6-5: Interpolation between projections of the same physical point..... 118
Equation 6-6: Constraint between the projections for planar case. 119
Equation 6-7: Distance function for planar case..... 119
Equation 6-8: Constraint between projections for cylindrical case. 120
Equation 6-9: Distance function for cylindrical case..... 120
Equation 6-10: Definition of disparity..... 123
Equation 6-11: Distance between viewpoint and testing ray for planar case. 124
Equation 6-12: Distance between viewpoint and surface for planar case. 124
Equation 6-13: Intersection searching function for planar case..... 124
Equation 6-14: Distance between viewpoint and testing ray for cylindrical case. 125
Equation 6-15: Distance between viewpoint and surface for cylindrical case. 125
Equation 6-16: Intersection searching function for cylindrical case. 125

Chapter 1
Introduction

Modeling and rendering are two central topics in computer graphics. As well, modeling is also an important topic in computer vision. As shown in Figure 1-1, conventional computer graphics focuses on the problem of synthesizing images from 3D geometric models, while computer vision concentrates on the inverse problem of recovering geometric models from images.

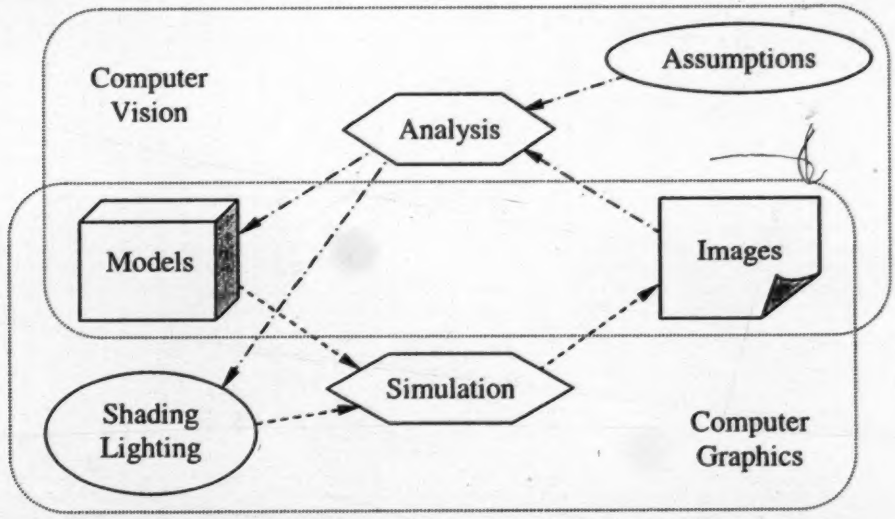


Figure 1-1: Domains of the traditional computer graphics and computer vision.

In Section 1.1, a brief introduction to the modeling and rendering processes is presented. The motivation of this work is discussed in Section 1.2. Section 1.3 gives the organization of the dissertation.

1.1 Consideration on Modeling and Rendering

In this section, models and images used in computer graphics and computer vision are introduced first. Then the modeling and rendering processes are discussed.

1.1.1 Models

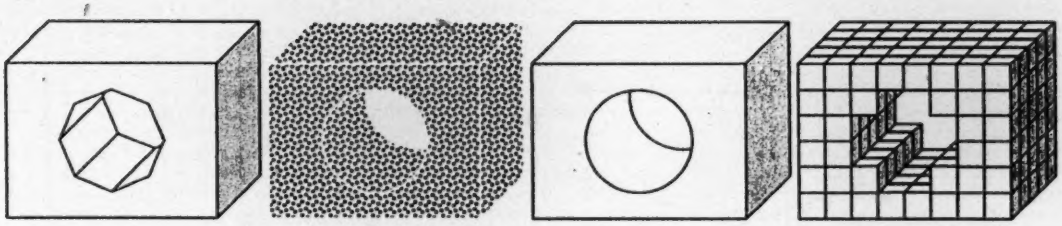
Models are used in both computer graphics and computer vision to represent objects in a scene. Commonly used models include the boundary representation, the point sample representation, the solid representation, and the discrete space representation.

In the boundary representation approach, the boundary of an object is described using surface primitives defined in 3D space. According to the primitives used, boundary representation can be further subdivided into polygon-based models, quadric models, and free-form models. A polygon-based surface model consists of polygonal facets, defined by the coordinates of their vertices. A quadric surface model consists of surfaces, defined by the parameters of corresponding functions. A free-form surface model consists of curved surface patches, defined by the coordinates of their control points.

The point sample representation uses points as primitives. It is a sampled version of the boundary representation. In this representation, objects are depicted using a set of points that lie on the boundaries of objects. Besides position, a point may contain additional geometric information, such as surface normal. The point sample representation is usually used in measurement results of real objects since the measurement is a sampling process.

The solid model representation describes objects using a Boolean combination of solid primitives to specify volumes, rather than surfaces. A complex solid model of an object can be assembled using Constructive Solid Geometry, by which the desired shape is specified in terms of a tree of solid primitives that are combined using basic set operations, such as union, intersection, and subtraction. The solid representation is widely used to represent manufactured parts since it naturally fits the manufacturing process.

The discrete space representation is a sampled version of the solid representation. It uses discrete volume elements, i.e. voxels, as primitives. In this representation, the space that contains the object is discretized into voxels. The object is then defined by a set of voxels that are inside the boundary of the object.



(a) (b) (c) (d)

Figure 1-2: The same object under different representations.

Figure 1-2 shows the same shape represented using four different models. In the boundary representation, Figure 1-2(a), the object is represented using fourteen polygons, eight of which are used to approximate the hole. In the point sample representation, Figure 1-2(b), points are used to describe the boundary of the object. In the solid representation, Figure 1-2(c), the object is defined by subtracting a cylinder from a box. Finally, in the discrete space representation, Figure 1-2(d), the object is depicted using 180 voxels.

1.1.2 Images

A continuous image is a function, i:$\mathbf{U}\rightarrow\mathbf{T}$ that maps from the parameter space, $\mathbf{U}\subset\Re^2$, to the multi-dimensional space $\mathbf{T}$. In this dissertation, the multi-dimensional space $\mathbf{T}$ is called the attribute space. A digital image, i_d:$\mathbf{U}'\rightarrow\mathbf{T}'$, is a sampled version of the continuous image. A digital image maps from samples in the parameter space, $\mathbf{U}'$, to a quantization of the attribute space, $\mathbf{T}'\subset\mathbf{T}$. Normally, the parameter space is sampled using an orthogonal uniform lattice:

$$\mathbf{U}' = \left\{\left(x_i, y_j \in \mathbf{U}\right) : x_i = i\cdot\Delta x, y_j = j\cdot\Delta y ; i, j \in Z ; \Delta x, \Delta y \in R^+\right\}$$

Equation 1-1: Sampling of the image parameter space.

For conventional images, the attribute space is one of the color spaces, such as RGB, HSV, I1'-I2'-I3', Lab, XYZ, and YIQ. In more recent approaches, the attribute space may be of higher dimensions and may carry information of other attributes such as transparency, depth, surface normal, etc.

The mapping from the parameter space to the vector space is normally one-to-one. This limitation can be relaxed to accommodate multi-layered images. In multi-layered images, multiple samples can be defined for the same pixel.

A camera model is sometimes associated with an image to specify the mapping between the 5D ray space[1] and the parameter space. Based on the lens used, camera models can be classified into the pinhole model, the thin lens model [131], the thick lens model [83], etc. Based on the shape of the image plane, the camera models can be further divided into the planar projection model, the cylindrical projection model, the spherical projection model, etc.

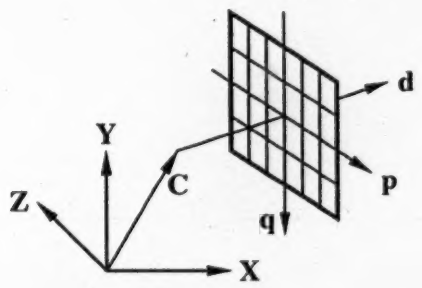


Figure 1-3: Planar pinhole camera model.

The planar pinhole camera model is the simplest and also the most popular camera model used in computer graphics and computer vision. As shown in Figure 1-3, a planar pinhole camera model is defined using the center of projection, **C**, viewing direction, **d**, and two basis vectors, **p** and **q**. Both basis vectors are perpendicular to the viewing direction **d**. Therefore, the image plane, which is spanned by the two basis vectors, is also perpendicular to the viewing direction.

1.1.3 Modeling

Modeling is an important topic in both computer graphics and computer vision. In computer graphics, modeling refers to techniques that can represent the shapes and appearances of objects photo-realistically. In computer vision, modeling is often referred to the appropriateness of models, reconstructed from images, as accurate representations of objects.

To describe natural objects using geometric models is very difficult, if not impossible. Many powerful approaches have been proposed to model natural objects such as soil

---

[1] Three dimensions are used to specify the starting location of the ray. The other two dimensions are used to specify the direction of the ray.

[99], stones [45], plants [44]; to simulate natural phenomena such as wind [173], clouds [122], fire [174], lightning [138]; and to model different parts of the human body such as hair [5], face [12], and muscle [24]. Nonetheless, using these approaches to create complex models requires a great deal of manual efforts and skills. Due to the variety of objects, it is not yet possible to provide a generalized modeling technique to model different kinds of objects. Therefore, the modeling process is very challenging and cumbersome.

Image-based modeling differs from traditional graphics modeling techniques in that the geometry and appearance of objects are derived from real photographs. In this sense, many techniques discussed in the computer vision area are often classified as image-based modeling techniques in the computer graphics area.

Image-based modeling techniques often result with shorter modeling times and more realistic models [41]. For example, suppose that we want to generate a virtual environment for the West Edmonton Mall. We could, according to the blueprint, build up the mall and decorate the stores in it. Although this could be done, it is nonetheless a laborious process. Alternatively, we could simply visit the mall and take photographs or even range images to reconstruct a visually acceptable model that provides enough information to generate novel views from arbitrary direction. The latter approach not only requires far less manual efforts, but also provides more realistic results, since photographs are, by definition, photorealistic.

1.1.4 Rendering

The pursuit of photorealism is one of the primary efforts of computer graphics research. Early 2D computer graphics approaches use wireframe models under perspective projection to give a sense of depth. Later on, hidden surface removal algorithms offer a more solid appearance. Shading algorithms allow surfaces to be rendered with varying brightness levels as if they were illuminated by sources of light [128], and shadow algorithms let objects to cast shadows on one another [182]. Photo-realistic renderings are often achieved using ray-tracing techniques, which can simulate specular reflections on shinny surfaces [181]. The development of radiosity enables the modeling of subtle effects of diffuse inter-reflections [55]. The path tracing technique tries to solve the

general rendering equation and can therefore create all optical effects [74]. More recently, the photon mapping technique is introduced as an efficient two pass global illumination method, which can render caustics as well as other optical effects [73].

The inputs to the above mentioned conventional rendering techniques are manually constructed scenes, which consist of geometric models, surface properties, and illumination information. Basically, the shading of an object is obtained through simulating the interactions of lights with the corresponding geometric model based on the properties of the material. The models are normally illuminated by light sources, and the image is captured using a virtual camera.

IBR is based on pre-rendered images or digitized images rather than geometric models. One of the main issues in IBR is to synthesize a novel view based on a set of images. Therefore, it is a signal reconstruction problem rather than a simulation problem. The most powerful features of IBR include the following:

- It is relatively inexpensive in term of computation cost;
- The cost of computing one frame is independent of scene complexity;
- It gives unprecedented levels of photorealism.

These features make IBR gain in popularity over the last few years. However, IBR does have the limitations in handling occlusions. Normally, to generate unlimited novel views for complex scenes, either dense samples or accurate depth information is needed to ensure that the computed images are physically correct.

1.2 Motivation

There are two main motivations of this work. The first one is to provide a framework for IBR techniques. The second one is to develop novel IBR techniques to handle dynamic scenes.

1.2.1 Taxonomy for Image-based Rendering

Although much research has been done in the IBR area, not much work has tried to provide a taxonomy of approaches [167] in this area. Such work would be helpful in the following aspects:

- To clarify the domain of IBR techniques. Due to the popularity of the term, many

approaches label themselves as IBR techniques. A more restricted definition of IBR is helpful to reduce the confusion.

- To answer the question — what is an image? The term "image-based rendering" itself is a bit of a misnomer, due to the rather loose definition of the term "image." A more general concept is, consequently, valuable to further the understanding.
- To demonstrate the relations among different IBR approaches. A systematical classification can illustrate the relations among them, which will promote novel approaches to be developed.

1.2.2 Dynamic Image-based Rendering

Little work has tried to apply IBR to dynamic scenes [80, 137, 187]. Dynamic IBR is an exciting new domain, with many potential important applications. Here, several existing and potential applications are listed in the following:

After populating the stadium with many cameras, we can record a soccer, hockey, or baseball game using multiple video sequences, each of which records the scene from a different viewpoint. Dynamic IBR techniques can then be used to generate video from any virtual viewpoint, no matter whether or not we have an actual camera at that location. Therefore, the audience can watch how the game progresses more clearly at an angle of their choice. Also we can use this technique to simulate a virtual camera that can automatically track the ball. So there will be no need to have a human to pan, tilt, or zoom a camera. A limited version of this idea has been implemented in the EyeVision system used for Super Bowl games [75].

Camera systems are widely used for security. In order to be able to look around, cameras are sometimes mounted on controllable platforms, which support the panning and tilting of the cameras. Unfortunately, changing the viewpoint is seldom supported. Therefore, some areas may not be visible because of occlusion. In addition, a single camera cannot face two different directions at the same time. If we could populate the walls of a museum with cameras, dynamic IBR techniques could be used to simulate as many virtual cameras as needed, each of which could be controlled separately to move around in the scene. The more cameras we use, the less occlusion will be experienced.

Like traditional computer graphics approaches, image-based modeling and rendering

techniques have been applied to the film industry as well. Currently these techniques are mainly used to create some special effects, such as the "bullet time" shots in movie "The Matrix" [40]. Dynamic image-based modeling and rendering can go one step further and make the movies of the future much more interactive. In making of these interactive movies, multiple cameras are used to record the actions from different viewpoints. When watching these movies, the audiences can choose viewpoint as they want, and move around in the scene as they wish. To find out who commits the murder, for example, they can make themselves to be in the right place, at the right time, and looking at the right direction. Of course, for the timid ones, they may also choose not to see it!

Video conferencing has become an attractive collaboration tool for geographically distributed teams as time and money could be saved on traveling. Most of the current video conferencing systems are window-based applications where videos of remote participants are displayed in separate windows. It is, therefore, not as effective as face-to-face communication since eye contact, gaze direction, and body language cannot be reproduced. Dynamic IBR techniques can be introduced into video conferencing so that an observer can view a remote participant from any viewpoint of choice [80]. As a result, in video-conferences of the future, the 3D appearances of remote participants can be captured and represented locally in your room. You can look at them from any viewpoint of choice, and therefore, the sense of presence can be enhanced.

1.3 Organization of the Dissertation

The rest of the dissertation is organized as follows: First of all, Chapter 2 reviews previous work in related areas, which include: image-based modeling, image-based rendering, stereo vision, image morphing, and image caching.

Chapter 3 presents the rayset concept, which is a novel taxonomy of IBR approaches. The rayset concept integrates different scene representations used in IBR. Based on this concept, existing IBR approaches are classified and links among them are demonstrated.

Three novel IBR techniques are discussed in Chapter 4, which are the object centered concentric mosaics, the spiral texture mapping, and the layer-based morphing. These approaches can be regarded as applications of rayset in static environments. Each approach has different requirements and advantages, and therefore, is suitable for

different applications.

Chapter 5 introduces a novel reliability-based stereo algorithm. This algorithm is used to estimate depth information, which helps to generate better rendering results. The experimental results show that the reliability-based stereo algorithm can accurately and efficiently estimate depth information.

In Chapter 6, a novel IBR approach, called camera field rendering, is presented. Using the depth information estimated with the reliability-based stereo algorithm introduced in Chapter 5, this technique produces sharp rendering results from sparse sampled views, which demonstrate that camera field rendering can be applied to dynamic scenes. That is, render novel videos based on sparsely sampled videos.

Finally, the dissertation concludes in Chapter 7, with discussions on future work.

Chapter 2
Previous Work

Many methods have been proposed in the image-based modeling and image-based rendering areas, both of which are strongly related to this dissertation. Therefore they are discussed in Sections 2.1 and 2.2 separately. Stereo vision techniques are often used to obtain depth information needed in image-based modeling and rendering algorithms. Hence, approaches in this area are covered in Section 2.3.

In this dissertation, IBR is defined as a class of techniques that is used to produce novel views of a scene or of an object using a collection of pre-acquired samples, without *a priori* knowledge of detailed geometric models of the scene or of the object.

Based on this definition, techniques such as image morphing and image caching are excluded from the IBR domain. The former one mainly focuses on generating a smooth transition between two different scenes and normally cannot be applied to generate novel views. The latter one assumes that the geometric model is known and uses images only for acceleration and not for representation. For completeness, approaches in these two areas are briefly discussed in Sections 2.4 and 2.5.

It is noteworthy that approaches such as view morphing employ morphing techniques to generate physically correct novel views [161]. Therefore, they are considered as IBR techniques in this dissertation.

2.1 Image-based Modeling

According to published information, previous work in the image-based modeling area can

be classified into four categories: models from silhouettes, models from color matching, models from range images, and models from human interaction.

2.1.1 Models from Silhouettes

Approaches in this category segment each input image into a binary object/background image. The silhouettes of an object are then used to construct a bounding volume of the object. The best volume that can be obtained from silhouette images is called the visual hull, which contains the real object [88].

Constructing a volumetric model of an object from multiple views is first described by Martin and Aggarwal [110]. Later on, many approaches have been proposed to construct an octree from three orthographic views [28]. These algorithms are not discussed here since they cannot be generalized for perspective images.

Reconstructing the volume of an object from multiple arbitrary perspective images is discussed in [64] and [130]. From each image, a 3D conic volume is formed using the silhouettes of the object and the center of projection. A model of the object is then constructed by intersecting conic volumes obtained from different images. Both approaches use the octree data structure to represent and to process the 3D volume data efficiently. Both approaches project the cubes of the octree onto the image plane to perform the object/silhouette intersection calculation.

Szeliski [175] enhances the above idea by constructing the model incrementally. Instead of building a separate octree for each view, he intersects each new silhouette with the existing model. In his approach, an octree representation of a cube that completely encloses the object is initialized. For each image, voxels at a coarse level of the octree are projected onto the image plane and tested against the observed silhouette of the object. If a voxel falls outside of the silhouette, it is removed from the tree. If it falls on the boundary, it is divided into eight smaller cubes, which are then processed recursively.

Kutulakos [86] recovers the surface of an object by computing the envelope of the light field [97] boundary. The rays defining the entire silhouette are mapped to points in an oriented projective space. The points are then approximated with curves whose envelope is consistent with all the image data. This approach does not rely on calibrated cameras or point correspondences.

Matusik et al. [111] model an object from its silhouette implicitly. In their image-based visual hull approach, the intersection between a novel ray and the implicit model of the object is calculated in the image spaces of the reference images. Since no explicit model is generated, this approach is fast enough to handle dynamic scenes in real time. The authors also claim that performing geometric computations in image space can eliminate the resampling and quantization artifacts.

Approaches in this category can generate complete (closed) surface efficiently and is fairly robust. A fundamental shortcoming is that they fail at concavities, where the model produced cannot enclose the object tightly. Additionally, volumetric approaches can only provide limited resolution due to the size of voxel.

2.1.2 Models from Color Matching

Approaches in this category [169] try to determine whether or not a point in the 3D space is occupied based on its projections on some reference images. The volume constructed is referred to as the photo hull. Under ideal conditions (the surfaces in the scene are Lambertian and there is no noise), the photo hull contains the true scene geometry and is a tighter fit than the visual hull.

In some approaches [33, 160], corresponding features in the reference images are determined. The output of these methods is a 3D representation of features. These approaches are not included in the review of this section since additional processes are needed for model reconstruction.

Seitz and Dyer introduce the concept of color consistency and use it to distinguish surface points from other points in a scene [162]. In their voxel coloring algorithm, the scene is represented as a finite set of opaque voxels, each of which has a fixed color. These voxels are traversed in depth-order and a voxel is added to the current set if it is consistent with the input images. In order to establish a fixed depth-order of points in the scene, this approach requires that no scene point be contained within the convex hull of the centers of cameras.

Saito and Kanade [142] apply the voxel coloring algorithm to the projective grid space. The projective grid space is defined by two basis views. The projection of points in the projective grid space on any other image can be calculated using only the

fundamental matrices between the image and the two basis views. Since the fundamental matrix can be obtained using several corresponding points in the images, the effort for Euclidean space camera calibration can be avoided.

The voxel coloring algorithm has restrictions on the camera locations. To allow for arbitrary camera placement, an extension, called the space carving algorithm, is proposed [87]. Unlike voxel coloring, space carving uses multiple scans, typically along the negative and positive directions of each of the three axes. During each scan, only cameras that have already been passed by the moving plane are used. This forces the scan to be from the near to the far relative to the camera used. Thus, when a voxel is evaluated, the transparency of other voxels that might occlude it from the cameras being used has been determined. Because carving is conservative, there is no need to add previously removed voxels back to the model as the algorithm runs.

For the scene point that is within the convex hull of the centers of cameras, the space carving algorithm only uses some of the cameras to determine its color consistency. Therefore, it is possible that color inconsistent voxels are included in the final model. Generalized voxel coloring [37] computes visibility exactly, and hence, yields a color consistent model. In two variants of the algorithm, two different data structures, called item buffers and layered depth images, are used to identify the images from which a voxel is visible. Their experimental results show that, compared with space carving, generalized voxel coloring can generate better visual reconstructions.

Inspired by the image-based visual hull approach, Slabaugh et al. implement the color consistency constraint in the image space. Instead of constructing the full photo hull as the above approaches do, the proposed image-based photo hull approach [170] produces only the portion of the photo hull that is visible to a virtual viewpoint. The approach starts from the visual hull found using Matusik et al.'s approach [111]. It then searches for color consistency points along the given novel ray in the image spaces of the reference images. Since it is conducted in the 2D image spaces, the reconstruction process is efficient.

Approaches in this category use the color information within the silhouettes of objects. Therefore, they generally need fewer images and can produce more accurate models than models from silhouettes approaches. However, these approaches assume that

the surfaces are Lambertian. Hence, they cannot be applied to scenes with specular objects.

2.1.3 Models from Range Images

Better models of objects can be reconstructed if the corresponding range images are available. Range images can be created using active optical range scanners, including structured light [10], triangulation or time-of-flight rangefinders [98], and focus range sensor [121]. Alternatively, they can be computed using computer vision techniques such as depth-from-stereo [148], motion, [69], shading [190], and texture [77].

The problem of modeling from range images for orthogonal views is first discussed in [29]. The kind of algorithms is not discussed here since they cannot be used for general perspective images. Some other approaches, namely unorganized point algorithms, reconstruct the model from 3D scatter points [65, 136]. They are not included either since the pixel relationship among range images is not employed.

Turk and Levoy [180] propose an incremental algorithm that constructs polygon meshes from multiple range images. In their algorithm, a triangulated mesh is created for each range image first. Meshes obtained from different range images are aligned with one another using a modified iterated closest-point algorithm. Overlapping portions of the meshes are removed, followed by connecting triangles along the remaining boundaries. Finally, discarded geometric information is re-introduced to establish final vertex positions. The drawback of this approach is that it may fail in areas of high curvature.

The volumetric integration algorithm [38] constructs surface models by integrating groups of aligned range images. For each range image, a continuous implicit function is defined, which represents the weighted signed distance of each point to the nearest range surface along the ray. Multiple range images are combined together so that a cumulative weighted signed distance function is defined for each voxel in the space. Therefore, the zero-crossings of such a 3D function define the surface implicitly. In their approach, the function is sampled on a discrete voxel grid and the marching cubes algorithm is employed to obtain the polygonal representation of the surface.

Rander et al. [137] apply volumetric integration in dynamic environments. The *virtualized* reality algorithm that they propose constructs a 3D representation of a scene

from multiple video streams. At a single time instant, a multi-camera stereo algorithm is used to compute range images for corresponding frames in different video streams. The range and intensity image pair is referred to as the visible surface model. The scene is modeled as a collection of visible surface models captured from different viewpoints at the same time. A complete surface model can then be generated using an adaptation of the volumetric integration technique.

Provided that the depth information is accurate, the models constructed by the above algorithms are better approximation to the real objects than both the visual hull and the photo hull. In addition, these algorithms employ the underlying structure of range images, and are therefore more robust than unorganized point algorithms.

2.1.4 Models from Human Interaction

More specific to certain applications, approaches in this category generate models from images with the aid of human interactions.

Debevec et al. [41] propose an interactive approach for modeling and rendering existing architectural scenes from a sparse set of still photographs. Their approach has two components, which are photogrammetric modeling and model-based stereo. Parameterized geometric entities, such as boxes, prisms, and surface of revolution, are used as modeling primitives. User inputs are needed to specify the shape and position of these primitives so that their projections align with their corresponding images in existing photos. Since the modeling process exploits the constraints that are characteristics of architectural scenes, it is convenient and effective. This approach is not suitable for modeling objects that are difficult to be represented by parametric polyhedral primitives.

Horry et al. [66] discuss how to make animation from only one image. With user interactions, the image is separated into background and foreground objects. The background in the scene consists of at most five rectangles, whereas polygons are used as a model for each foreground object. This technique makes it possible to "walk or fly through" a single picture. Because of the lack of information, the approach does not and cannot solve the problem of recovering areas that are occluded by foreground objects. In their approach, commercially available 2D paint software is used and user interaction is needed.

2.2 Image-based Rendering

Many approaches have been presented in the IBR area [78] and they can be classified by several methods. According to how much geometric information is used, they can be classified into rendering without geometry, rendering with implicit geometry, and rendering with explicit geometry [167]. Furthermore, according to how the sampling is done, they can be classified into view-centered approaches and object-centered approaches.

In this dissertation, IBR approaches are classified according to how images are used. This forms the following five categories: images as textures, images as references, images as layers, images as primitives, and images as datasets. Generally speaking, approaches in the first and the fourth categories are object-centered approaches. Approaches in the second and the third categories are view-centered approaches. Approaches in the last category can be used for both object-centered and view-centered applications.

2.2.1 Images as Textures

Texture mapping [61, 62] is the earliest IBR approach. Approaches in this category use images to represent part of the object. These images are mapped onto surfaces of coarse geometric models to add fine photometric details.

Two dimensional texture mapping is proposed by Catmull [21] to generate color variance on surfaces. Later on, it is used to generate other attributes of surfaces including specular reflection [15], roughness [13], and transparency [49]. Aono [6] generalizes these approaches using the concept of attribute mapping.

Bump mapping [14] is the first approach to use images to modify the geometric properties of the underlying models. It simulates the appearance of wrinkled surface by performing perturbations to surface normals. Although it produces realistic images, it cannot simulate the self-occlusion effects among bumps since the surface itself is not modified. Hence, it can model only bumps whose heights are negligibly small as compared to the extent of the associated surface.

Displacement mapping [34] adds depth information into images in order to specify the amounts by which a desired surface deviates from the underlying model. Hence, it

alters not only the surface normals of but also the shapes as well. The displacement maps are rendered using micro-polygons in Cook's original approach. Later, other rendering approaches are proposed using ray-tracing [104], image warping [152], and multi-layer texture mapping [81].

Relief texture mapping [126] can be considered as an extension of displacement mapping. It uses image warping to handle the parallax and visibility changes that result from the 3D shape of the surface. In this approach, a relief texture is first converted into an ordinary texture using 1D warping along rows and columns separately. The resulting texture is then mapped onto the polygon using standard texture mapping techniques.

The above approaches use one image to record the appearance of the surface from a single direction, which is usually perpendicular to the surface. As a result, non-diffuse illuminations are not captured. View-dependent texture mapping [41] composites together textures obtained from multiple views based on the observer's viewpoint. It can simulate geometric details more realistically, since it better represents non-diffuse reflectance and can simulate the appearance of unmodeled geometry. In their later work, Debevec et al. [42] use visibility preprocessing, polygon view maps, and projective texture mapping [159] to reduce the computational cost and to produce smoother blending.

The view-dependent texture mapping approach requires only a few photos of the surface to be textured. It is easy to use, but may not be able to fully capture the view-dependent illuminations of the surface. The surface light field approach [186] is proposed to address this problem. It captures the appearances of the surface from all possible directions and resamples the data using an intermediate representation. The experimental results show that this approach can be used to construct virtual images of shiny objects under complex lighting conditions.

In view-dependent texture mapping and surface light field, when rendering a ray in the novel view, the contributions of different input images are calculated based on the angular differences only. Buehler et al. argue that the resolution difference between the novel view and a given input view is also an important factor. In their unstructured lumigraph approach [18], the weights of different input views are calculated based on both angular penalties and resolution penalties of these views. As a result, when the same

object shows up in multiple input images with different resolutions, this approach can choose the proper input images and blend them together.

Rather than compositing the input views directly, the dynamic texture mapping approach [30] uses a set of basis textures. These basis textures are generated based on all input images using Principal Component Analysis. To render novel views, new textures are synthesized by modulating the basis textures. Rendering can be done in real time using hardware acceleration. However, this approach is limited to small visibility changes, and therefore, the user can observe the object from a limited viewing range only.

Texture mapping techniques still rely on geometric models to provide the coarse shapes of the object to be rendered. In addition, it is often difficult to specify the mapping functions, which map 2D images onto 3D surfaces. Texture mapping is also known to be subject to the aliasing problem. Filtering techniques, such as mip-maps [183] and summed-area tables [36, 53], can be used to address this problem.

2.2.2 Images as References

Approaches in this category use images rather than geometric models to represent a scene. New views of the scene are generated through interpolating existing reference images.

The Movie-Map system [100] is one of the earliest attempts in this category. In this approach, thousands of reference images are stored; each of which can be accessed randomly. The system can also pan, tilt, or zoom the stored images to fit the required viewpoint. As an example application, the streets of the city of Aspen are sampled at 10-foot intervals, and four cameras are used to shoot the views at every point.

Instead of using real pictures, the Virtual Museum [117] is based on pre-rendered synthetic images. At each selected point in the museum, 45 views are rendered to form a 360-degree panning movie, which allows the user to look around. The path between the connected points is also rendered to form a bi-directional transition movie, which simulates working from one point to the other in both directions.

QuickTime VR system [26] uses cylindrical panoramas rather than several images to sample the scene at each selected point. The cylindrical panorama of a scene can be

created using a specially designed panorama camera, or by rotating a regular camera and stitch together the images. Since the panoramas are "orientation-independent," this approach offers the user smooth 2D rotation capability at selected viewpoints. However, it does not allow the user to move freely in the scene.

To introduce more freedom, Shum and He propose the concentric mosaics approach [166]. This approach employs multiple cylindrical mosaics, each of which samples the scene from a certain distance away from the center of a cylindrical manifold. The concentric mosaics for a scene can be created by simply spinning an off-centered camera on a rotary table. When rendering a novel view, data from different mosaics are reused and combined together. This approach allows the user to move freely in a circular region with a reasonably small file size. Nonetheless, it only provides parallax effects in the horizontal direction but not in the vertical direction.

The parallax effect can be generated using the depth image approach [27], which stores the depth information at every pixel. After the depth is introduced, the backwards mapping function that maps from the output sample location to the input image is no longer available. Forwards mapping can be exploited, but one has to handle the problems of overlaps and gaps. The gaps between samples arise when a surface is either magnified or uncovered in the novel view, while the overlaps occur when a surface is either covered or contracted. In Chen and Williams's approach, gaps are filled by interpolating the colors of adjacent pixels and view-independent visible priority is used to solve the overlap problem.

The overlap problem can be handled more efficiently using the occlusion compatible ordering algorithm [115], which is essentially a painter's style algorithm. In this algorithm, the projection of the output viewpoint on the input image plane, i.e. the epipole, is computed. If the output viewpoint is behind the input viewpoint, then pixels closer to the epipole are warped first. Otherwise, pixels that are farther to the epipole are warped first. This warping order guarantees that pixels that map to the same location in the output image are painted in the back-to-front order.

Simply interpolating the color of adjacent pixels to fill gaps is efficient. Nonetheless, it may cause artifacts because the warping process does not preserve image topology, i.e. the adjacency among pixels. Mark et al. [107] propose two solutions, which are the splat-

based and mesh-based reconstructions. The first approach maps a single point in the input image to multiple pixels in the output image. The second approach treats the input depth image as a mesh of micro-polygons, and then scan-converts these polygons in the output image. However, both approaches have limitations. Splatting will cause unnecessary blurring and it cannot guarantee that all the gaps are filled. Meshing will generate rubber sheet effects between the foreground and background objects.

Several approaches [63, 156] extend the idea of mesh-based reconstruction by creating a mesh model in world coordinates. In their approaches, the reference images' centers of projections are projected onto the desired novel view. A triangulation process is performed using these centers as vertices. Depth information is used to compute the depths of the vertices in the image space of the novel view. A recursive subdivision process may be applied to make the triangle mesh closely represents the true geometry of the scene. Rendering can then be done by texture mapping the triangles.

After the depth information is introduced, the backward mapping function cannot be applied directly to compute the corresponding point in the input image for a given sample in the output image. However, one can search for the corresponding point using the epipolar constraint [82]. This constraint states that for a given pixel p in one of the reference images with the center of projection C_1, the corresponding pixels in another image with the center of projection C_2 must lie on its epipolar line, which is the projection of ray C_1p on image C_2. The projection of C_1 on image C_2, i.e. the epipole, is the intersection of different epipolar lines.

Based on this observation, Laveau and Faugeras [89] suggest performing backwards mapping through searching suitable pixel in the input image for each output sample location. Their approach does not require fully calibrated views. Instead, weakly calibrated views are employed. However, the computation costs for searching the suitable pixel is obviously high.

Satoh et al. [145] also generate new views using backwards mapping. They use 7×7 cameras that lie on a plane to capture the scene. The Stereo by Eye Array [146] algorithm is used to compute the disparity information for each 3×3 camera subset. Assuming that the disparity maps obtained are correct, their algorithm can generate arbitrary view by searching for consistent pixels in existing views.

Seitz and Dyer [161] notice that linear interpolation is shape preserving when both reference image planes are parallel to the transition line, which is the line connecting the two projection centers. Hence, under this circumstance, physically correct in-between views can be generated using morphing techniques. Based on this observation, a three step approach, called view morphing, is proposed. In the first step, the reference images are pre-warped to make their image planes parallel to the transition line. Then existing morphing techniques are applied to generate the in-between views. Finally, post-warp is used, if necessary, to adjust the viewing directions.

If the disparity map for two reference images is available, the disparity morphing algorithm [68] can be applied. In their paper, Huang et al. deduce the equations for interpolating corresponding points under several different scenarios. For the forward transition, both reference images are pre-warped first such that their optical axes are collinear with the transition baseline. Then a non-linear equation is used to interpolate between corresponding points. For the lateral transition, their result shows that the interpolation equation becomes linear, this agrees with the view morphing result.

The above approaches do not require 3D geometric models of the scene. Since these approaches do not sample the scene sufficiently, the number and quality of reference images limits the quality of reconstructed images.

2.2.3 Images as Layers

Approaches in this category use an image to represent part of the scene as a layer. Different approaches can be used to generate different layers and the whole scene can be composited with these layers according to their relative positions.

An environment map is a projection of a scene onto a specific geometric shape, such as a cube [57], a sphere [15], or a cylinder [26]. The environment map is initially used to simplify the computations of specular reflections on shiny objects. Here, it can be used as the background layer since it is perfect for simulating objects at a great distance from the camera, such as the sky. Environment mapping is invariant to translation. As a result, it cannot produce parallax effects.

Sprites are texture maps with alphas (transparency) rendered onto planar surfaces [129]. They are widely used in video games as drawing primitives. In IBR, this technique

is used to simulate unimportant foreground objects. Although sprite can display some parallax effects relative to other objects, it cannot simulate internal parallax, i.e. parallax effect within the sprite itself.

Using the idea of depth images, one can add depth information to sprite to generate sprite with depth [164]. Shade et al. propose a novel idea to render sprite with depth, which is to separate the warping process into two steps, namely, a parallax warp and a perspective warp. Sprite with depth can display internal parallax but cannot deal with uncovering of occlusion due to depth discontinuity. Hence, it is suitable for modeling objects with smooth varying depth only.

Based on the observation that pixels at similar depth move in a similar way during warping, Schaufler proposes the layered impostors algorithm [151]. A layered impostor consists of more than one transparent polygon. The image of parts of the object, which are at a similar distance to the viewer, is mapped to one of the polygons. Essentially, layered impostor can be considered as another representation of sprite with depth, even though the two approaches are proposed independently. Since multiple textures are used, rendering is more efficient through hardware acceleration. In addition, layered impostors algorithm can handle disocclusions better by drawing slightly overlapping depth intervals onto every layer.

To fully deal with disocclusions, one needs to know the color information of surfaces that are not visible in the reference image. The layered depth image approach [164] can be used here. A layered depth image is a view of the scene from a single camera view, but it stores more than one sample along each line of sight. Since the visibility problem can be handled, layered depth images can be used for more complex geometric objects.

The layered depth image tree approach [23] combines hierarchical space partitioning scheme with the concept of layered depth image. The motivation is to preserve the different sampling rates of different reference images. A layered depth image tree is an octree with a layered depth image attached to each octree node, and all layered depth images having the same resolution. The layered depth image tree is constructed by warping each pixel of the reference images to a layered depth image of an octree cell at appropriate level, then filtering the pixels to the layered depth images of all ancestor cells in the octree. During rendering, a layered depth image at an appropriate level is used for

each pixel in the output image.

Approaches in this category are more flexible since they can be used to model individual scene element. Different approaches can be used together through composition. However, they are still viewpoint-centered approaches. This limits the desired view to stay inside a certain neighborhood of the reference image's center of projection.

2.2.4 Images as Primitives

Approaches in this category can fully represent an object. Therefore, they can be used to model a scene, together with other image-based or geometry-cased primitives.

The easiest way to represent an object is to use Object Movie [26]. An Object Movie is a 2D array of frames, each of which corresponds to a viewing direction. This representation contains redundant information and still cannot provide parallax effects.

The Delta Tree [39] is a data structure that represents an object using a set of reference images. All the reference images are taken from a point on a sampling sphere that encloses the object. The Delta Tree encodes these images in a quad-tree that divides the angular space. Each node of the Delta Tree stores an image taken from a point. The image is compressed by discarding pixels that can be reconstructed by warping its ancestors' images to its viewpoint. Therefore, the Delta Tree construction ensures that no surface of the object is sampled more than once. The rendering process can create novel views by traversing the Delta Tree.

Schaufler [150] uses layered impostors to represent an object. Different viewpoints are selected from the sphere that encloses the object. In order to ensure an even distribution, the vertices or centers of sides on a platonic solid, such as dodecahedron, are used. For each viewpoint, a number of depth-augmented images are generated as a layered impostor. This approach also uses the so called back perspective projection, instead of the traditional perspective projection, to sample a larger portion of the object's surface per image.

The view-based rendering approach [132] also uses multiple range images to render real objects. In their approach, each input dense range map is replaced by a sparse triangle mesh that closely approximates it. Each triangle mesh is then texture mapped

using the associated color image. To synthesize an image of the object from a novel viewpoint, triangle meshes constructed from three closest viewpoints are individually rendered, and the rendering results are blended together with a pixel-based weighting algorithm using software Z-buffering.

Simply using a collection of images may result in some area to be poorly sampled. In addition, combining information from different images poses some difficulties since the space of the viewpoint is not sampled continuously. Rademacher and Bishop try to alleviate these problems using multiple-center-of-projection images [133]. In their approach, the camera moves along a path smoothly and samples a 1D strip each time. Both color and depth information is recorded for each pixel, and so is the camera information for each column. In the rendering stage, pixels are re-projected and Z-buffered to produce the desired view. This approach is also capable of sampling different portions of a scene at different resolutions.

The occlusion compatible order does not exist among images taken at different centers of projection. Therefore, the depth information needs to be Z-buffered to resolve the visibility problem. In order to reduce the computation cost, Oliveira and Bishop present the image-based object approach [125] using the idea of back perspective projection. An image-based object consists of six layered depth images defined on the six faces of a cube. These layered depth images share a single center of projection, which is the geometric center of the cube. As a result, an image-based object can be rendered with the correct perspective from any viewpoint using a list-priority algorithm. For synthetic objects, the image-based object representation can be easily generated using a modified ray tracer that keeps all intersections along a ray. For real objects, a laser range finder is required to sample the object and the samples obtained are reprojected onto the six faces of the cube.

The above approaches represent objects in the 3D world space. Therefore, the computational cost for rendering is relatively high. To solve the problem, several approaches are proposed to do rendering in the image space of the reference views. The pioneering work in this area is done by Matusik et al. Their approach [111] renders the novel view of the object in the scene based on the constructed view-dependent visual hull. Later on, Slabaugh et al. [170] render the object using the view-dependent photo

hull, which encloses the object more tightly than the visual hull. More recently, the opacity hull has proposed and used by Matusik et al. [112] to handle highly specular and fuzzy materials. The opacity hull is defined as the visual hull of the object with the view-dependent opacity information extracted using matting techniques. Since the opacity information provides soft boundaries, this approach can seamlessly blend fuzzy objects, such as fur and feathers, into new environments.

Approaches in this category are object-centered approaches. The objects they modeled can be viewed from arbitrary direction. Therefore, they can be used to represent foreground objects that surround the viewer.

2.2.5 Images as Datasets

Approaches in this category use rays rather than images as the underlying sampling primitives. Therefore, the difference between object-centered approaches and viewpoint-centered approaches are eliminated. These approaches can be cast as attempts to fully sample the plenoptic function over a subset of space.

The original plenoptic function [1] defines the intensity of rays of any direction, from any location in space, at any time, and over any range of wavelengths. By ignoring time and wavelength, McMillan and Bishop [116] define the plenoptic model as a 5D function, which maps any ray in space to a color vector. In their approach, a set of panoramic images at different locations is used to represent the scene.

In light field rendering [97], rays are represented by their intersections with two planes in arbitrary position, one of which may be placed at infinity. Such a pair of planes is called a light slab. The plane that the centers of projections are on is denoted as the camera plane, and the common image plane is called the focal plane. Light fields can be created from both rendered images (virtual environment) and digitized images (real environment). After it is created, an image with a given imaging geometry can be generated by sampling a 2D slice of it. To make creation, transmission and display of light fields practical, a two-stage compression scheme is used, which includes fixed-rate vector quantization and entropy coding.

Gortler et al. [56] use a set of six light slabs arranged as a cube, called the lumigraph, to enclose an object. When reconstructing the desired view, the lumigraph algorithm uses

an approximate geometric model of the object to improve the rendering quality. For real objects, the model is created using a volume from silhouette technique. This approach also allows using arbitrary camera poses to construct the lumigraph, whereas in light field rendering, the camera needs to be carefully controlled to scan along a plane. Since the samples provided by arbitrary camera poses are not evenly spaced, a pull-and-push algorithm is used to fill the gaps. In the pull stage, a lower resolution approximation is derived using samples inside a wider area. During the push phase, the lower resolution approximation is used to fill in regions in the higher resolution that do not have enough samples.

Both the light field rendering and lumigraph approaches try to do rendering in real time. However, both have limitations in different ways. Light field rendering operates pixel by pixel, and therefore, is sensitive to image resolution. Lumigraph takes advantage of texture-mapping hardware, but is limited by the texture memory. To address these limitations, Sloan et al. [171] propose several methods to accelerate the rendering process by trading off quality for time.

Another limitation of light field rendering and lumigraph approaches is the dense sampling rate requirement, resulting that an enormous amount rays need to be sampled and stored. Warping-based lumigraph rendering approach is proposed to address this problem [154, 155]. In their approach, according to the position of the novel view, a region of interest is determined for each reference image. Pixels in the region are forward warped based on per-pixel depth information. Results from different reference images are then blended together to generate a single physically correct texture for the novel view. This approach can handle sparse samples. However, it seems to rely on accurate per-depth information. No results for real scene are given in [154], and the preliminary results in [155] show that faulty depth information reduces the rendering quality considerably.

Under the light field rendering concept, sampling schemes other than the two-plane parameterization also have been studied. The spherical light field approach [70] uses spherical coordinates to parameterize the 4D space. This scheme can be considered as a collection of small directional spheres hanging to a large positional sphere, which is the convex hull of the bounded object. The two spheres are tessellated using polyhedrons. The discretized spherical data are reordered into a 2D array and are compressed using

nonstandard Haar wavelet.

Camahort et al. [20] also use a tight bounding sphere to enclose the object. However, their approach emphasizes on how to uniformly sample the rays that intersect the sphere. Two different parameterizations are used, which are two-sphere parameterization and sphere-plane parameterization. In their approach, the most complex platonic solid, the icosahedron, is used as the generator to subdivide the sphere.

Polar parameterization scheme is also proposed [179]. For any given ray, this scheme first locates the point on the ray that is closest to the center. Then the coordinates of the ray are defined using the distance between the point and the center, the two polar angles for the direction of the point, and the rotation angle of the ray within the tangent plane. This parameterization scheme has the advantage in that it gives a compact generic representation for rays in the ray space. The authors also claim that the rendering results produced by the polar parameterization are likely more visually pleasing than those generated using six light slabs.

The parameterization scheme can also be changed interactively during rendering. In the dynamically reparameterized light field approach [72], a dynamic focal surface is used. This gives the user additional degrees of freedom, which can be used to produce effects such as changing the depth-of-field and the apparent focus. This approach can also be used to render moderately sampled light fields. It generates sharp rendering results for objects that are close to the focal surface, even though objects far away from the focal surface may appear blurry. More recently, a real-time distributed rendering system is proposed based on this algorithm [187].

In all of the above approaches, the plenoptic model is reduced to four dimensions by assuming that the space is transparent and that the region of interest is free of occluders. This assumption helps to reduce the storage requirement. However, it also limits these algorithms to be used in free space. Lischinski and Rappoport [101] place the light field rendering approach and the layered depth image approach [164] within a common framework to handle non-free space cases. The novel idea is to represent simultaneously but separately both view-independent and view-dependent scene information. View-independent part is described using a layered depth cube, which consists of three orthogonal high-resolution layered depth images. View-dependent part is modeled

through layered light field, which consists of a large collection of low-resolution layered depth images, each of which captures the appearance of the scene from a particular direction.

Approaches described in this section densely sample a scene or an object. Since view-dependent lightings are captured, they can generate non-diffuse effects, such as highlight and inter-reflection. The drawback of these approaches is that they require too much storage even after compression.

2.3 Stereo Vision

The input of stereo vision is two or more images taken by parallel cameras. In the case the cameras are not parallel, image rectification [7] is applied to align the image planes so that they are coplanar and their scanlines are horizontally or vertically collinear. The output of stereo vision is a disparity map or disparity maps corresponding to multiple input images. Stereo vision differs from image-based modeling since it only produces discrete view-dependent depth information of the scene.

Stereo vision algorithms can be coarsely classified into feature-based approaches and intensity-based approaches. Feature-based approaches [59, 123] try to detect and then match features. The output of these methods is sparse depth points. These approaches are not included in the review of this section since interpolation is needed to obtain a complete disparity map.

The intensity-based approaches [148, 176] are reviewed in the rest of this section. These approaches are further classified into the following categories: local methods, cooperative methods, optimization methods, and multi-view methods.

2.3.1 Local Methods

For each pixel, local methods select a window. Pixels within the window are used to compute the matching score or matching cost between input images. The disparity that can produce the best match is set as the disparity of the pixel.

The selection of window size is a critical problem for local methods. The window size should be large enough to include enough intensity variation for reliable matching. On the other hand, the size should be small enough to avoid disparity variation inside the

window. The generated disparity maps will be noisy if the widow size is too small and blurry if the widow size is too large. Levine et al. [96] change the window size locally according to the intensity pattern.

Kanade and Okutomi believe that the optimal window size depends not only on intensity variation but also on the disparity variation, which is an unknown itself. Therefore, they propose an iterative algorithm [76]: starting from an initial estimate of the disparity map, the algorithm iteratively updates the disparity value for each pixel by choosing the size and shape of window until it converges.

The position of the window is also important. The above two approaches always use windows centered at the pixel where disparity is calculated. This introduces a large disparity variation within the window when the pixel is close to an occlusion boundary. This problem can be alleviated using shiftable windows [47]. In their approach, for each pixel, nine windows anchored at different locations are used to calculate the matching costs, and the window that gives the smallest matching cost is used. In addition, the authors also use left-right consistency to detect occlusions and to remove mismatches.

Local approaches are very efficient, and many of them can produce disparity maps in real time. However, these approaches compute disparities based on local information only and are therefore sensitive to noise.

Some researchers have investigated how to detect and remove errors from disparity maps [106, 144]. Instead of trying to find a corresponding pixel for every pixel in the source image, these approaches reject ambiguous correspondences. Consequently, in the resulting disparity map, only pixels with unambiguous matches have disparities assigned.

2.3.2 Cooperative Methods

Cooperative methods apply global constraints in the matching process. The disparity of a pixel is influenced by disparities of its neighbors. Iterative processes are used to update disparities until a stable state is reached.

The work by Marr and Poggio [108] is one of the first to apply global constraints to produce disparity maps. Two assumptions about stereo, namely, uniqueness and continuity, are stated explicitly and used as global constraints. The first assumption states that every pixel has a unique disparity value. The second one states that disparity values

are generally continuous except across occluding boundaries. Based on these two assumptions, the authors believe that a pixel should support its neighbors who have similar disparity values. A simple cooperative algorithm is presented for diffusing the supports among different pixels.

Later on, the stereo problem is formulated as the process of extracting a surface from a 3D *u-v-d* volume, i.e., the so-called disparity space [71, 188]. The value of any voxel in the disparity space denotes the probability of the disparity of pixel (u,v) is d. Consequently, the desired disparity map should be a surface that satisfies two constraints: (1) the surface is smooth; (2) it passes through voxels of high probabilities. This formulation is interesting in that it naturally implements the continuity assumption.

Different algorithms are proposed under this formulation. In Chen and Medioni's approach [25], the disparity surface is extracted from the volume using a propagation type algorithm. The algorithm consists of two steps: seed voxel selection and surface tracing. Time efficiency is achieved by executing the algorithm in a coarse-to-fine fashion.

In Zitnick and Kanade's approach [194], a 3D function is used to update the volume iteratively so that the possibilities of different disparities can influence each other. After the process converges, the disparity with the maximum possibility is selected as the disparity for corresponding pixel. Zitnick and Kanade also suggest that occluded areas can be explicitly detected by setting a threshold. If the maximum possibility value is smaller than the threshold, the pixel will be classified as occluded.

Since global constraints are applied, cooperative algorithms are more robust with respect to noise than local methods. However, they may not converge at the global optimum solution.

2.3.3 Optimization Methods

Approaches in this category use global optimization techniques to search for the best solution in terms of a given global cost function. The main distinction between these algorithms is the minimization procedure used, which includes simulated annealing [51], dynamic programming [4], and graph cut [17].

Stereo vision is first considered as one of the ill-posed problems in computer vision

by Marroquin et al. [109]. Ill-posed problems denote problems that do not have sufficient information to solve. These problems can be formulated as optimization problems using the smoothness constraint. In Marroquin et al.'s approach, the ill-posed problems are solved using simulated annealing. Later on, Barnard also tries to solve the stereo vision problem using simulated annealing under the Markov Random Field framework [9]. In theory, simulated annealing can find the global minimum if run for long enough. However, it is generally too slow to converge.

If the smoothness constraint is only enforced along scanlines, a 1D energy function can be formulated. Dynamic programming can then be applied to find the global minimum for each scanline in polynomial time. Under dynamic programming, occlusions can be detected explicitly using the monotonicity or ordering constraint [50]. This constraint requires that the relative ordering of pixels on a scanline remain the same between the two views.

Dynamic programming approaches suffer from the difficulty of enforcing inter-scanline consistency. As a result, horizontal streaks will show up in the generated disparity maps. This problem can be alleviated through either minimizing inter-scanline discontinuities [35] or using ground control points to ensure that the solution path goes through highly reliable matches [16]. The ground control points can also help the solution path to make large disparity jumps that might otherwise have been avoided because of large occlusion costs.

In addition, enforcing the ordering constraint also has limitations because it does not hold when narrow foreground objects exist in the scene. Scanline optimization tries to solve this problem by using a different strategy to enforce the smoothness constraint [148]. In their approach, a smoothness term is used and a disparity value is assigned at each pixel such that the overall cost along the scanline is minimized.

For a Markov Random Field problem with a label set of size two, the global minimum can be found using a minimum cut algorithm [58]. As a result, it is possible to solve problems that involve multiple labels using multiple passes of graph cuts. Based on this observation, Boykov et al. propose an iterative algorithm, which uses two types of moves, namely expansion moves and swap moves, to solve the stereo vision problem [17]. These moves can simultaneously change the labels of arbitrarily large sets of pixels,

and therefore, the iterative algorithm can converge much faster. Later, this technique is extended by Kolmogorov and Zabih so that occlusions can be detected [84].

In general, compared with approaches in the previous two categories, optimization methods can produce more accurate disparity maps since they put efforts on finding the global minimum.

2.3.4 Multi-view Methods

Recovering disparity information from two views has its inherent limitations. First of all, the length of the baseline, i.e. the distance between the two cameras, has to be carefully chosen. Using a short baseline cannot estimate distance accurately due to the narrow triangulation. Using a longer baseline, however, not only increases the range in disparity to be searched for a match but also the possibility of a false match. Hence, a tradeoff has to be made between precision and correctness. Another limitation is that, in a complex scene, some part of the scene may be visible in only one of the views. As a result, the disparity information is not recoverable because there is no correct match.

Okutomi and Kanade try to solve the first limitation using a multiple-baseline approach [124]. They place multiple cameras along a line. All of the cameras are aligned, with their optical axes perpendicular to the camera plane. A matching cost is computed for each stereo pair. The algorithm then minimizes the sum of different matching costs obtained from multiple stereo pairs. The key idea of this approach is relating the matching cost to the inverse distance of the image point to the actual object rather than relating the matching cost to the disparity. The former measure is unique because there is only one true inverse distance value for each pixel, whereas for the latter, the disparity value is proportional to the length of the baseline and varies for different stereo pairs.

Stereo by Eye Array [146] is proposed to solve the second limitation. In this approach, a 3×3 array of cameras are arranged on a plane, with the same baseline length between neighboring cameras on both the vertical and horizontal directions. Eight stereo pairs, each of which consists of the center image and one of the peripheral images, are used to calculate matching costs. Occlusions are distinguished from false targets based on the knowledge of typical combinations of occluded cameras. Eight camera masks are defined assuming that occluding boundaries are locally linear. In their later work [119],

new masks are defined without the assumption and the camera matrix is increased to 5×5. They also make a systematic comparison of different algorithms used for occlusion detection in the Stereo by Eye Array approach [147]. The result shows that the sort-and-weight algorithm can give relatively accurate results in both areas with and without occlusion.

When handling the same dataset, Kolmogorov and Zabih [85] use five images (center, top, bottom, left, right), forming either four or ten stereo pairs. Instead of calculating an overall cost, the matching costs obtained from different pairs are used separately. For a given smoothness term, the graph cut technique is used to find a local minimum that is close to the global minimum. Since an optimization technique is used, this approach is much less sensitive to noise. However, the computational cost is much higher than that of the Stereo by Eye Array approach.

Approaches in this category use multiple images to solve the two inherent problems of binocular stereo vision, namely, the precision-and-correctness problem and the occlusion problem. In general, the disparity maps generated using these approaches are more accurate than those generated by binocular stereo matching algorithms.

2.4 Image Morphing

Image morphing [185] is a technique used to transfer and transform one digital image into another. It has been proven to be a powerful tool for visual effects in the entertainment industry. Problems related with morphing include feature specification, warping generation, and transition control.

The distinct feature of image morphing is that it aims at interpolating between images of two different scenes, rather than creating novel views for a single scene. Hence, the objects shown in the in-between images are not necessarily real objects. This is the reason that these approaches are not included in the IBR category.

Related work in volume morphing [95] and image warping [103, 140] are not included in the review of this section. The former technique is a 3D extension of image morphing. The latter one requires only one reference image and change the shape (and color) of the object in the scene during the computation.

2.4.1 Feature Specification

Different from cross dissolving, which interpolates the colors of reference images only, image morphing applies both 2D geometric transformations and color interpolations. Normally, user interactions are needed to specify corresponding features in reference images. In existing morphing algorithms, meshes, line segments, and point sets are used.

Meshes are used in the pioneer morphing approach proposed by Smythe [172]. This approach has been used in Ron Howard's 1988 movie "Willow" and many subsequent motion pictures. In this approach, meshes are defined and adjusted manually in both reference images so that landmarks lie below the corresponding grid lines. The two meshes are constrained to be topologically equivalent, i.e., no folding or discontinuities are permitted. Using meshes for feature specification is straightforward, but it needs lots of user interactions.

This feature-based morphing approach [11] has been successfully applied in Michael Jackson's 1992 music video "Black or White". Line segments are used in this approach. Corresponding landmarks in the reference images should lie within the corresponding line segments. The feature specification process is simplified. However, line segments cannot intersect each other, otherwise unexpected distortions will appear.

Any 2D geometric primitive can be point sampled, which is a more general form of feature specification. Following this idea, feature point sets are used by Lee et al. [90], Litwinowicz and Williams [102], and Ruprecht and Mueller [141]. In addition, Lee et al. [91] also adapt the snake technique to reduce the burden of feature specification.

Generally, feature specification is cumbersome. Researchers attempt to reduce the need for user interactions needed for this process. For two similar images, the algorithm presented by Gao and Sederberg [48] can be used to generate morphing with little or no user interactions. Their algorithm seeks to minimize the work, which is defined as a function of the amount of warping and recoloration needed to deform one image into the other. A hierarchical method is used to find a minimal work solution among all possible warps.

2.4.2 Warping Generation

After the feature correspondences are specified, they are used to compute the warping

function that defines the mapping between points in the reference images. The way feature specification used usually limits the choice of the warping function that can be applied.

In mesh warping [172], bicubic interpolation is used to determine the mapping of points not lying on the mesh. The algorithm is efficient since the warping function is locally defined.

In feature-based morphing, line segments are used to specify the features. A pair of corresponding line segments in the reference images defines a coordinate mapping. When two or more segment pairs are specified, a weighted sum of all segments' effects is calculated. The speed of this algorithm is somewhat slow since all segments are referenced for every pixel. Lee et al. [93] propose an optimization method based on piecewise linear approximation to accelerate the process.

When the specified features are a set of corresponding points, generating the warping functions for the x- and y-components is equivalent to constructing two surfaces that interpolate scattered points. Based on this idea, Lee et al. [90] and Litwinowicz and Williams [102] propose two similar warping techniques that employ the thin plate surface model.

Also using corresponding points as specified features, Lee et al. [91] propose the multilevel free-form deformation to achieve C^2-continuous and one-to-one warps among feature point pairs. In the paper, the cubic B-spline surface is used to define the free-form deformation and a sufficient condition for a 2D B-spline surface to be one-to-one is presented.

2.4.3 Transition Control

Transition control determines the rate of the geometric transformation and color interpolation process. Usually, linear (uniform) transition functions are applied. Nonetheless, more interesting results are possible if different transition rates are used for different parts. For example, transition control can produce effects that some features are transferred first, while the deformation of other features can be delayed.

When meshes are used for feature specification, transition control can be implemented by assigning transition curves to all mesh nodes. When point sets are used

for feature specification, transition control can be done by selecting a set of points and by specifying the transition curves for them. The points selected for transition control does not necessarily relate to the points used in feature specification [91].

Transition control can also be implemented using procedural transition functions [91]. The authors demonstrate the morphing result when a linear function varying in the vertical direction is applied. A convincing transformation is generated, in which one input image transfers into the other from top to bottom.

2.4.4 Other Related Work

The morphing algorithms discussed so far consider only two reference images at a time. Morphing among multiple images is implemented by transforming from one image to the next one in a sequence. Lee et al. [92] present a general framework that supports morphing among multiple images. In this framework, each input image is considered to be a vertex of a convex polyhedron in a multi-dimensional space and the in-between images are considered to be a point inside the convex polyhedron.

Traditional morphing techniques can only generate transformation from one image to another. More recently, the light field morphing technique [193] is proposed to produce transformation between two light fields. This approach uses 3D feature polygons as control features to specify the ray correspondence in the 4D ray space. Ray-space warping is also introduced to fill arbitrarily large holes caused by object shape changes. The experiments show that this approach can produce convincing 3D morphing effects.

2.5 Image Caching

Image caching approaches can be applied to synthetic images only — either pre-rendered or dynamically generated [43]. They all assume that the full 3D geometric model of the object or the scene is known. Another distinct feature of these approaches is that images are used for accelerating the rendering process only, and not for simplifying the modeling process. Therefore, these approaches are not classified in the IBR category.

According to how to cache and what to cache, image caching approaches can be classified into pre-rendering, dynamic updating, and sampling plus shading.

2.5.1 Pre-rendering

Approaches in this category replace the geometric models in the scene with pre-rendered textures so that simplify models are small enough to be rendered in real time. Therefore, these approaches can also be regarded as one of the model simplification techniques, together with two other techniques, which are the visibility culling algorithm [177] and the level-of-detail algorithm [46].

Basically, pre-rendering approaches separate the using of geometric-based models and image-based models in spatial space. To be precise, geometric models are used for nearby (important) objects and pre-rendered images are used for faraway (unimportant) objects.

The concept of impostor is introduced by Maciel and Peter [105]. An impostor is defined as an entity that is faster to draw than the true object, but retains the important visual characteristics of the true object. In their approach, parallel projected images with different viewing directions are pre-rendered and used as view-dependent impostors. In their visual navigation system, these impostors are used to represent objects or clusters of objects to maintain high and approximately constant frame rates. However, since parallel projection is used, which implies that the viewing distance is infinite, these image impostors cannot be used to simulate objects that are close to the viewer.

Simply using textured polygons to replace an arbitrary subset of the model may cause discontinuity when the viewpoint changes. That is, the geometry surrounding the texture does not match the geometry sampled in the texture. This problem can be solved through image warping, provided that the depth information is added to the texture. Nonetheless, if one warps the texture to match the geometry, it has to be done pixel by pixel and for each frame. To avoid this, Aliaga [2] warps the geometry to match the texture so that C^0 continuity can be maintained. Aliaga also discusses how to perform smooth transitions between geometry and texture by morphing the nearby geometry. The drawback of this approach is that the views generated are not physically correct.

In architectural walkthrough applications, models are often subdivided into cells and portals so that visibility culling techniques can be applied. Aliaga and Lastra [3] improve this cells and portals framework by adding image simplification, i.e. using textures at portals as replacements for the geometry. Later on, this idea is extended further by using

depth images [134] and layered depth images [135] instead of the conventional texture. In the latter two approaches, pre-rendered depth images or layered depth images are warped to the current viewpoint so that physically correct views can be generated.

For urban environments, visibility culling techniques cannot effectively reduce the size of the model to be rendered. For this kind of scene, Sillion et al. [168] suggest segmenting the model into local regions based on natural urban subdivisions. In their system, a 3D impostor is used to model regions far away from the current viewpoint. To generate such an impostor, distant models are rendered first to create a depth image. Then a constrained triangulation is performed inside the external contour of the image. Since depth information is used, a 3D impostor has longer "cache life" than a single textured polygon.

The advantage of pre-rendering approaches is that it is possible to deal with potentially unbounded geometric complexity in a preprocessing step. The disadvantage is that they use a fixed representation to depict the scene. Therefore, the point of view must stay within a limited region for which this representation is generated.

2.5.2 Dynamic Updating

Approaches in the category locally cache the results of slower rendering and then generate new views by warping. Basically, they separate geometric-based models and image-based models in the temporal space. Some frames are generated by rendering the geometric models and some are produced by warping the dynamically cached images.

Regan and Pose [139] propose a hybrid rendering system, which applies dynamic image caching. The geometry-based subsystem renders scene onto the six faces of a cube centered at the viewer location at available rendering rate. Once the image cache is generated, the image-based subsystem can update the scene at interactive rate when the user changes the viewing direction. In order to better exploit coherence, they also segment the scene according to the distance to the viewer and treat fast moving foreground objects and slow changing background separately.

Schaufler [149] employs dynamically generated impostors in his real-time rendering system. When rendering the scene, objects are replaced by image impostors. Since the depth information is discarded, the polygon's flatness becomes apparent when the

viewpoint changes. In Schaufler's approach, the images of objects are dynamically updated to let impostors closely resemble the original objects.

Two similar algorithms [153, 165] are proposed independently to improve Regan and Pose's approach by using a hierarchical set of images. The hierarchical structure introduced can effectively reduce the number of images required. In both algorithms, the scene is encoded in a binary space partitioning tree by placing splitting planes inside the gaps between objects. During the walkthrough of the scene, images stored in nodes at various levels are reused. If the viewpoint moves too much, these images are updated. Shade et al. also discuss error estimation and cost-benefit analysis, which are used to decide whether or not to cache an image.

The Talisman architecture [178] is proposed to reduce the computation cost of high quality animation. In this architecture, the renderer renders animated objects each into independent image layers at available rate, if necessary. The warper applies affine transformations to cache image layers in order to simulate 3D motion of objects using translation, rotation, scaling, and skewing. Finally, the compositor combines all the layers in real time in each frame.

In the layered rendering approach [94], not only different objects but also different shadings of a single object are separated. The underlying idea is that different parts of a scene may have different requirements on spatial and temporal sampling rates. So, dividing them into layers can take advantage of the temporal and spatial coherence. For example, highlights from fast moving lights and shadows caused by fast moving objects could be rendered at a higher frame rate and at a lower resolution than stationary objects.

Most of the dynamically updating approaches take advantage of texture-mapping hardware so that cached images can be displayed in real time. These approaches also have the advantage that the rendering results can be progressively refined to the correct image, i.e. the one that would have been obtained using the original geometry. However, the drawback is that the potentially unbounded complexity of the geometry may overwhelm the capability of the rendering sub-system.

The shortcomings of the pre-rendering and dynamically updating approaches can be overcome by combining them into a single framework [43]. Their approach uses pre-calculated images if necessary, and transits to dynamically generated images for better

quality when time allows.

2.5.3 Sampling plus Shading

The rendering process itself can also be separated into a sampling process and a shading process. This idea is supported by observations that the sampling process, i.e. the intersection computation, is normally time-consuming and that the shading process can be efficiently done in real time.

In Max and Ohsaki's approach [114], an object is sampled using parallel projected Z-buffers, which are pre-computed for a number of preset viewing directions on the unit sphere. The Z-buffers store multiple depth levels at each pixel, allowing hidden objects to be reconstructed. They also store encoded normal and color information, so that the shading can be done as a post process. More recent work by Max [113] uses a hierarchy of reference images of differing spatial resolution. However, Max and Ohsaki do not address the problem of how to reconstruct continuous surfaces. They try to eliminate holes by super-sampling the objects.

In point sampled rendering [60], an object is sampled by orthographic views. In order to obtain uniform sampling, an equilateral triangle lattice rather than a square lattice is used. Color, depth, and normal information are stored for each point sample, enabling Z-buffer composition, Phong shading, and shadow effect. A pyramid of Z-buffers is used to detect gaps in the output image. A variation of the pull-and-push algorithm is used to fill all the gaps.

Since shading is done in real time, both approaches support dynamic lighting. However, they use a fixed representation to depict objects and can therefore provide only limited level of detail.

Chapter 3
Taxonomy for Image-based Rendering

McMillan and Bishop claim that "all IBR approaches can be cast as attempts to reconstruct the plenoptic function from a sampled set of that function" [116]. However, their framework does not specify how the plenoptic function is sampled. Chai et al. [22] analyze how many samples of the plenoptic function are needed to reconstruct a new scene. Nonetheless, their derivation is based on the two-plane parameterization used in light field rendering only, and they do not analyze other sampling schemes.

In addition, the question of what is an image [32] has not yet been answered. Conventionally, an image is considered as a 2D RGB valued function. Recently, depth images and multiple-center-of-projection images have been proposed for IBR. The question is: Are they still images? In addition, 4D light fields, lumigraphs, and 3D concentric mosaics are also introduced. So, what are their relations with conventional images?

In this chapter, a novel concept, *the rayset*, which is more general than the concept of an image, is presented. It is claimed in this dissertation that different image-based scene representations can all be cast as different kinds of raysets. The scene reconstruction techniques used in IBR, therefore, can be regarded as attempts to render different raysets.

In the rest of this chapter, the definition of rayset is given in Section 3.1 first. Different scene representations are cast as different raysets in Section 3.2. Section 3.3 classifies different scene reconstruction techniques using the rayset taxonomy. Two different ways of scene editing are discussed in Section 3.4.

3.1 Rayset Taxonomy

The plenoptic function [1] defines the intensity of ray in direction (θ,ϕ), from point (x,y,z) in space, at time t, and with a wavelength λ:

$$p = P(x, y, z, \theta, \phi, \lambda, t)$$

Equation 3-1: Original plenoptic function.

For a static scene, and consider only the visible light domain, the original plenoptic function can be reduced to five dimensions, which maps any ray in space to a color vector [116]:

$$\mathbf{C} = P(x, y, z, \theta, \phi)$$

Equation 3-2: 5D plenoptic model.

The plenoptic model is an explicit function that depicts the scene. Clearly, a parametric function can also be used instead. This inspires the definition of a rayset.

3.1.1 Definition of Rayset

A rayset is a parametric function, which is defined as:

$$\begin{cases} (x, y, z, \theta, \phi) = \mathbf{S}(\mathbf{U}) \\ \mathbf{T} = \mathbf{A}(\mathbf{U}) \end{cases}$$

Equation 3-3: Definition of rayset.

where **U** is the parameter space, **T** the attribute space, **S(U)** the support function, and **A(U)** the attribute function.

The support function maps from the parameter space to the 5D ray space. It defines how to sample the scene. The attribute function maps from the parameter space to the attribute space. It defines the sampling results. The dimension of the parameter space is called the dimension of the rayset.

Different IBR approaches can be cast as attempts to sample and reconstruct the scene using different raysets. All these approaches need to address the following two problems:

- How to sample a given scene using rays? That is, what kind of image-based scene representation is used?
- How to reconstruct the scene based on the sample rays? That is, how to render the

image-based scene representation and what is the additional information needed?

In the next two subsections, the approaches in IBR area are classified according to how they address the above two problems, i.e., the scene representation used and the scene reconstruction technique used.

3.1.2 Classification of Scene Representations

In terms of how the scenes are sampled, it is clamed in this dissertation that different image-based scene representations can all be cast as different kinds of raysets.

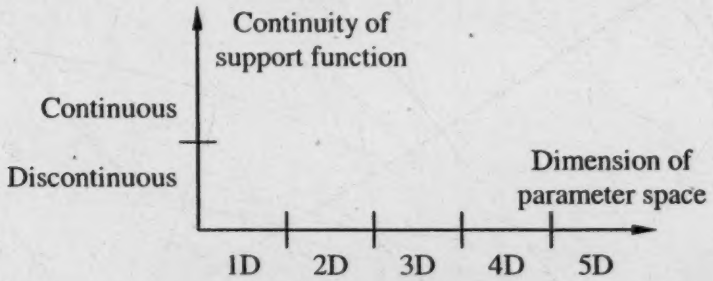


Figure 3-1: Classification of raysets.

As shown in Figure 3-1, raysets can be classified according to the dimension of the parameter space. Contingent on whether or not the support function is continuous, raysets can also be classified into continuous raysets and discontinuous raysets.

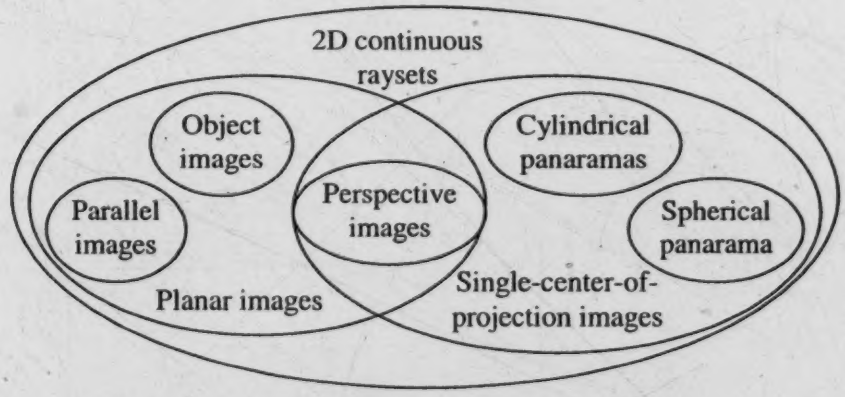


Figure 3-2: Classification of 2D continuous raysets.

As shown in Figure 3-2, according to the type of the projection surface, 2D continuous raysets can be further divided into planar images and non-planar images, the latter of which includes cylindrical panoramas and spherical panoramas. According to the type of the center of projection, 2D continuous raysets can be divided into single-center-

of-projection images and multi-center-of-projection images, with the latter one includes parallel images and object images. The perspective images belong to both planar images and single-center-of-projection images.

3.1.3 Classification of Scene Reconstruction Techniques

Now let us see how to classify different scene reconstruction techniques in IBR area under the rayset taxonomy.

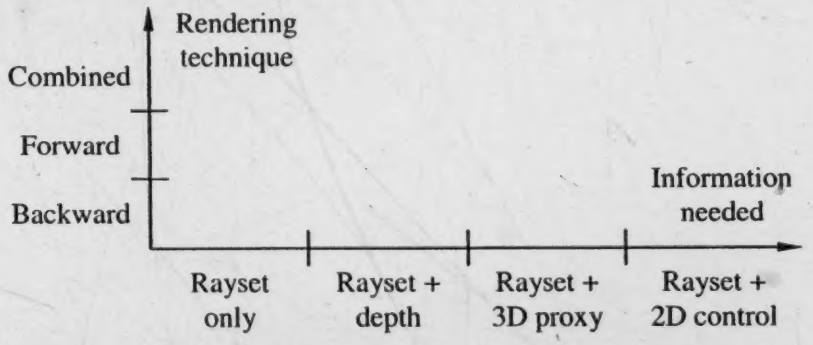


Figure 3-3: Classification of scene reconstruction techniques.

As shown in Figure 3-3, different scene reconstruction techniques can be classified based on either the rendering approach used or the information needed. According to the rendering technique used, they can be classified into backward techniques, forward techniques, and combined forward backward techniques. In the backward technique, for a given testing ray, its parameters are computed first through a reverse process based on the support function. The attributes of the ray can then be found directly using the attribute function. In the forward technique, all the rays defined in the rayset are enumerated. Each ray is reprojected to the desired view and the attributes of the ray are used to color its corresponding ray in the desired view.

It is noteworthy that the direct inverse function of the support function may not exist since the parameter space normally has a lower dimension than the ray space has. The reverse process is accomplished under some assumptions. For example, in light field rendering, it is assumed that the space is transparent and that the region of interest is free of occluders. Therefore, rays that start from different locations along the direction of the ray are considered the same.

According to information used in the rendering process, scene reconstruction

techniques can be classified into the following approaches: approaches using rayset only, approaches that need depth information, approaches that require 3D proxies, and approaches that need 2D control information.

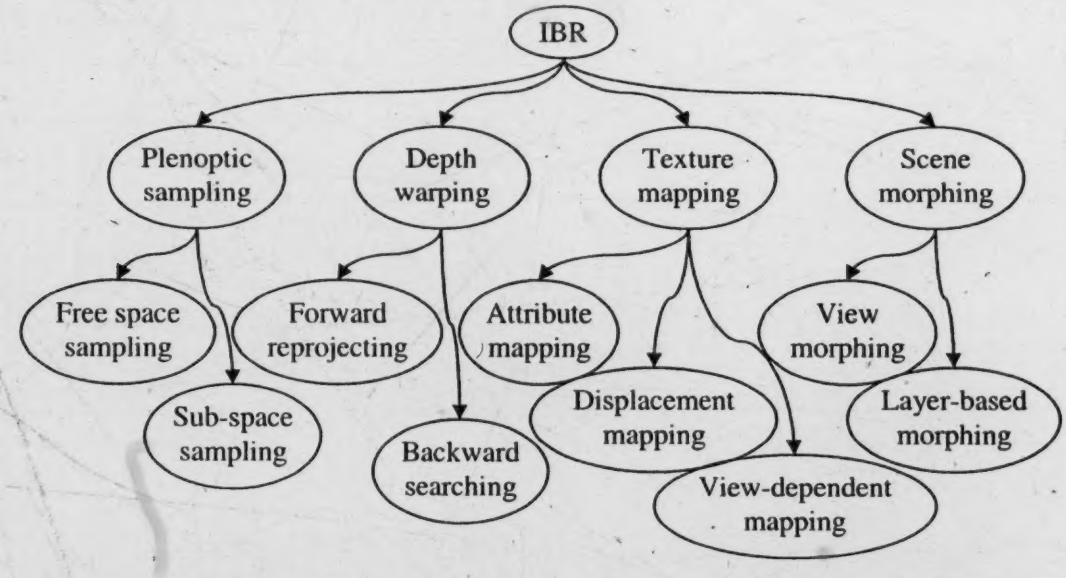


Figure 3-4: More detailed classification of scene reconstruction techniques.

In this dissertation, these four categories are called: plenoptic sampling, depth warping, texture mapping, and scene morphing. A more detailed classification based on these four categories is shown in Figure 3-4.

3.2 Rayset Examples

Most IBR techniques employ some kinds of 2D representations. Many of these approaches use the simplest planar images. In Subsection 3.2.1, planar images are cast as different 2D raysets first. Other 2D representations are covered in Subsection 3.2.2. Representations of higher dimensions are cast as raysets of corresponding dimensions in Subsection 3.2.3.

3.2.1 Planar Images

A planar image has a single image plane. It is generally called an image in the literature. Here planar images are further classified into three types according to the form of projection used.

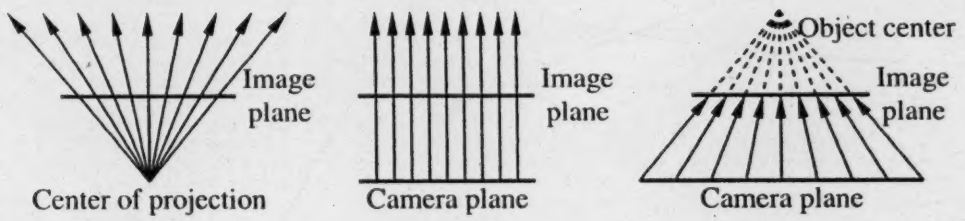


Figure 3-5: Planar images (a) perspective image (b) parallel image (c) object image.

As shown in Figure 3-5(a), perspective images are images obtained using the planar pinhole camera model. All the rays defined in a perspective image start from the center of projection, a.k.a. the viewpoint. Based on the planar pinhole camera model shown in Figure 1-3, the support function of corresponding rayset representation can be defined as:

$$\begin{cases} \mathbf{S}_x(s,t) \equiv \mathbf{C}_x \\ \mathbf{S}_y(s,t) \equiv \mathbf{C}_y \\ \mathbf{S}_z(s,t) \equiv \mathbf{C}_z \\ \mathbf{S}_\theta(s,t) = \Theta(\mathbf{d} + s \cdot \mathbf{p} + t \cdot \mathbf{q}) \\ \mathbf{S}_\phi(s,t) = \Phi(\mathbf{d} + s \cdot \mathbf{p} + t \cdot \mathbf{q}) \end{cases}$$

Equation 3-4: Support function for perspective images.

where, $\mathbf{C}_x$, $\mathbf{C}_y$, and $\mathbf{C}_z$ are the x-, y-, and z-coordinates of the center of projection, $\mathbf{d}$ is the viewing direction, $\mathbf{p}$ and $\mathbf{q}$ are the basis vectors, Θ and Φ are the functions that calculate θ and ϕ of the ray:

$$\Theta(\mathbf{v}) = \arctan 2(\mathbf{v}_y, \mathbf{v}_x)$$
$$\Phi(\mathbf{v}) = \arcsin(\mathbf{v}_z / \|\mathbf{v}\|)$$

Equation 3-5: Functions for converting vectors to angles.

Perspective images represent the scene that is visible from a given viewpoint. It is the most commonly used representation in IBR. Figure 3-6 shows the relationship among different representations derived from perspective images.

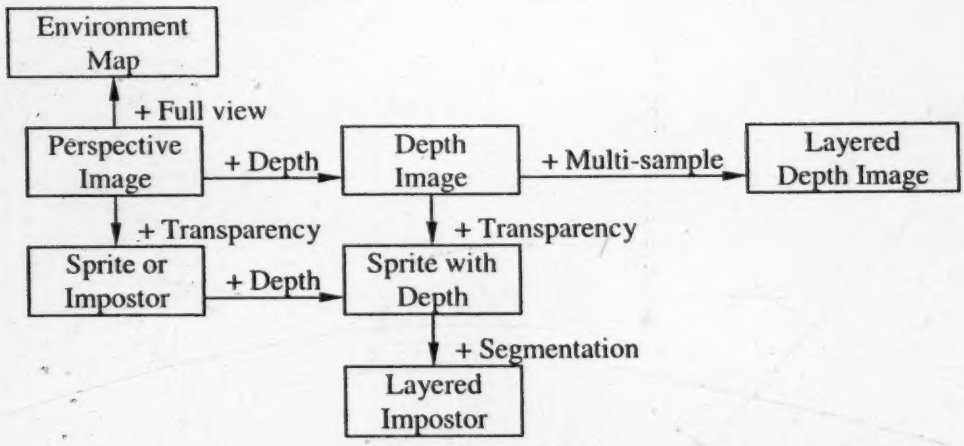


Figure 3-6: Representations based on perspective images.

Images acquired using a scanner are parallel images. As shown in Figure 3-5(b), all rays defined in a parallel image are parallel to the viewing direction. Different rays start from different locations on a plane, which is referred to as the camera plane. Assuming that the parametric equation of the camera plane is $\mathbf{F}(x,y)$, then the support function of the corresponding rayset representation can be defined as:

$$\begin{cases} \mathbf{S}_x(u,v) = \mathbf{F}_x(u,v) \\ \mathbf{S}_y(u,v) = \mathbf{F}_y(u,v) \\ \mathbf{S}_z(u,v) = \mathbf{F}_z(u,v) \\ \mathbf{S}_\theta(u,v) \equiv \theta_0 \\ \mathbf{S}_\phi(u,v) \equiv \phi_0 \end{cases}$$

Equation 3-6: Support function for parallel images.

where (θ_0,ϕ_0) specifies the viewing direction, $\mathbf{F}_x$, $\mathbf{F}_y$, and $\mathbf{F}_z$ the x-, y-, and z-components of function $\mathbf{F}$.

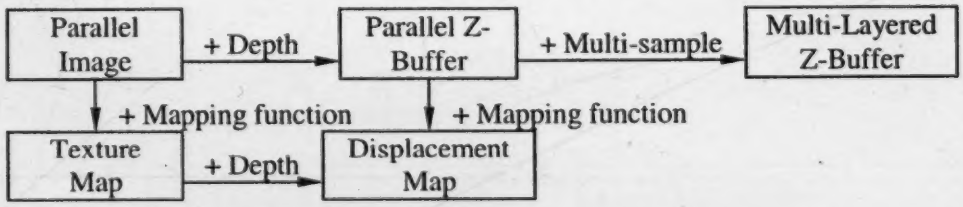


Figure 3-7: Representations based on parallel images.

Parallel images represent the appearance of a scene viewed from an infinite distance. A parallel image has the merit that it samples the object more evenly, while a perspective

image has bias to objects that are closer to the viewpoint. Several representations used in IBR that are derived from parallel images are shown in Figure 3-7.

As shown in Figure 3-5(c), different rays defined in an object image also start from different locations on the camera plane. However, all these rays pass through the same point in the scene, which is denoted as the object center. Assuming that the parametric equation of the camera plane is $\mathbf{F}(x,y)$, then the support function of corresponding rayset representation can be defined as:

$$\begin{cases} \mathbf{S}_x(u,v) = \mathbf{F}_x(u,v) \\ \mathbf{S}_y(u,v) = \mathbf{F}_y(u,v) \\ \mathbf{S}_z(u,v) = \mathbf{F}_z(u,v) \\ \mathbf{S}_\theta(u,v) = \Theta(\mathbf{O} - \mathbf{F}(u,v)) \\ \mathbf{S}_\phi(u,v) = \Phi(\mathbf{O} - \mathbf{F}(u,v)) \end{cases}$$

Equation 3-7: Support function for object images.

where $\mathbf{O}_x$, $\mathbf{O}_y$, and $\mathbf{O}_z$ are the x-, y-, and z-coordinates of the object center.

It is noteworthy that object images are different from object movies. An object movie keeps the appearance of an object viewed from different points on a sphere and is a 4D representation. An object image is a planar projection image and is a 2D rayset.

Object images have been used in IBR research. However, they are sometimes referred to as perspective images, which may cause misunderstanding. For example, Oliveira and Bishop [125] refer to object images sharing the same object center as images sharing the same center of projection. Schaufler [150] uses the name "back perspective images" to denote object images. However, he does not distinguish the difference between the object center and the center of projection.

For clarifications, here the differences between an object image and a perspective image are highlighted:

- In an object image, all testing rays start from different positions. Therefore, they are actually multiple-center-of-projection images. In a perspective image, however, rays start from the same location, which is the center of projection.
- An object image is normally used to sample the outside of an object. A perspective image is normally used to sample the inside of an environment. Therefore, when only one intersection exists between the testing ray and the scene, object images and

perspective images sample different sides of the same surface.

- When multiple intersections exist, a perspective image records the attribute of the intersection that is closest to the center of projection. But an object image records the attributes of the one that is closest to the camera plane, which is normally the furthest one from the object center.

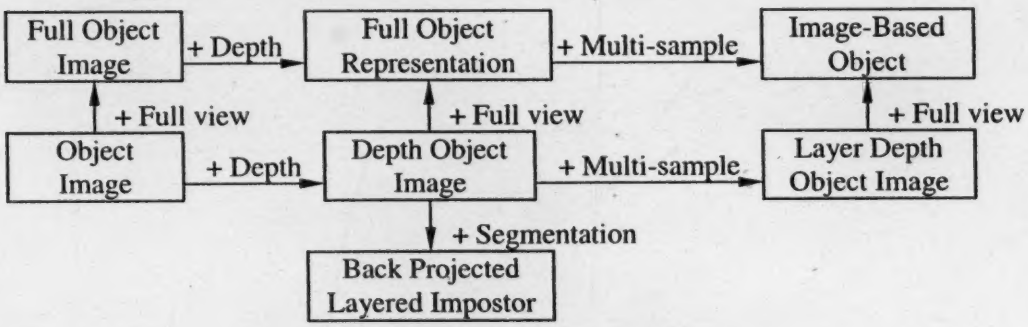


Figure 3-8: Representations based on object images.

The rays defined by an object image do not converge. Therefore, an object image does not describe a physically correct view. However, if an object image contains depth information, it can be re-projected to create new views. As shown in Figure 3-8, representations based on object images are mainly used for approximating objects.

3.2.2 Non-planar Images

Many other IBR approaches employ 2D representations other than planar images, such as cylindrical panorama. These representations can be classified as some kind of 2D raysets.

A cylindrical panorama can be captured using special devices that rotate around the camera's optical center. Cylindrical pinhole-camera model is often used to map pixels on a cylindrical panorama to the ray space. To classify cylindrical panorama into 2D rayset, the support function can be defined as follows:

$$\begin{cases} \mathbf{S}_x(u,v) \equiv \mathbf{C}_x \\ \mathbf{S}_y(u,v) \equiv \mathbf{C}_y \\ \mathbf{S}_z(u,v) \equiv \mathbf{C}_z \\ \mathbf{S}_\theta(u,v) = u \\ \mathbf{S}_\phi(u,v) = \arctan 2(v,f) \end{cases}$$

Equation 3-8: Support function for cylindrical panoramas.

where, $\mathbf{C}_x$, $\mathbf{C}_y$, and $\mathbf{C}_z$ are the x-, y-, and z-coordinates of the center of the projection, and f is the focal length, which is also equal to the radius of the cylinder.

The concept of multiple-center-of-projection images is similar to that of 2D continuous raysets. Both of them allow different samples to have different viewpoints and both require the viewing rays to vary continuously across neighboring samples. Nonetheless, they are not exactly the same since a multiple-center-of-projection image defines the mapping between the parameter space and the ray space implicitly using a set of cameras, while a 2D continuous rayset describes the mapping relation explicitly using a support function.

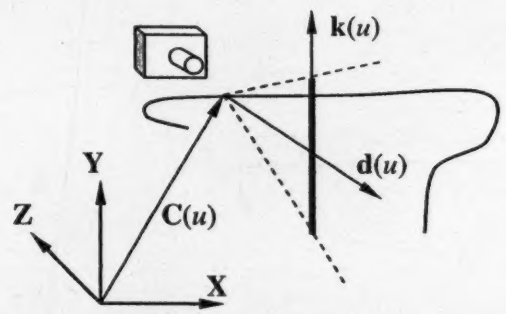


Figure 3-9: Strip camera model.

Rademacher and Bishop [133] use images taken by a strip camera as examples of multiple-center-of-projection images. Figure 3-9 shows the strip camera model. Assume that the camera path is defined using a curve, whose function is $\mathbf{C}(u)$, the viewing direction is defined using function $\mathbf{d}(u)$, and the direction of the strip is defined using function $\mathbf{k}(u)$. Then the support function of the corresponding rayset representation is defined as:

$$\begin{cases} \mathbf{S}_x(u,v) = \mathbf{C}_x(u) \\ \mathbf{S}_y(u,v) = \mathbf{C}_y(u) \\ \mathbf{S}_z(u,v) = \mathbf{C}_z(u) \\ \mathbf{S}_\theta(u,v) = \Theta(\mathbf{d}(u) + v \cdot \mathbf{k}(u)) \\ \mathbf{S}_\phi(u,v) = \Phi(\mathbf{d}(u) + v \cdot \mathbf{k}(u)) \end{cases}$$

Equation 3-9: Support function for images taken by a strip camera.

3.2.3 High-dimensional Representations

High-dimensional representations are also utilized in previous work. These

representations can be cast as raysets with corresponding dimensions. Actually, the plenoptic model itself can be expressed as a 5D rayset. Such a rayset is just the parametric representation of the plenoptic model.

Both the light field and lumigraph approach use light slabs as scene representations. Light slabs parameterize rays using four parameters. Hence, they can be cast as 4D raysets.

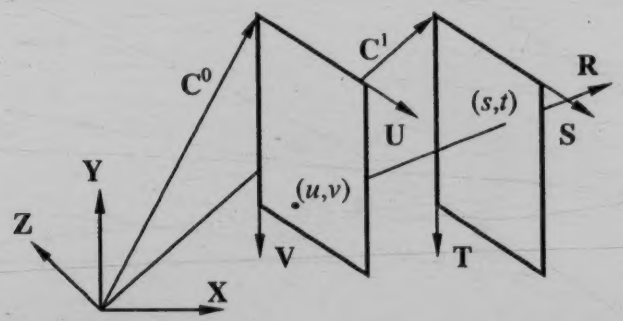


Figure 3-10: Two-plane parameterization used in light field rendering.

A light slab parameterizes rays by their intersections with two planes. Following the notation used in light field rendering, the camera plane is denoted as the UV plane, and the focal plane as the ST plane. As shown in Figure 3-10, the parameters (u,v,s,t) define a ray, **R**, that starts from a point on the UV plane whose local coordinates are (u,v), and passes through another point on the ST plane whose local coordinates are (s,t). Suppose the parametric equations of the UV plane and the ST plane are $\mathbf{F}(x,y)$ and $\mathbf{G}(x,y)$, respectively. Then, the support function of the corresponding rayset representation can be defined as:

$$\begin{cases} \mathbf{S}_x(u,v,s,t) = \mathbf{F}_x(u,v) \\ \mathbf{S}_y(u,v,s,t) = \mathbf{F}_y(u,v) \\ \mathbf{S}_z(u,v,s,t) = \mathbf{F}_z(u,v) \\ \mathbf{S}_\theta(u,v,s,t) = \Theta(\mathbf{G}(s,t) - \mathbf{F}(u,v)) \\ \mathbf{S}_\phi(u,v,s,t) = \Phi(\mathbf{G}(s,t) - \mathbf{F}(u,v)) \end{cases}$$

Equation 3-10: Support function for light slabs.

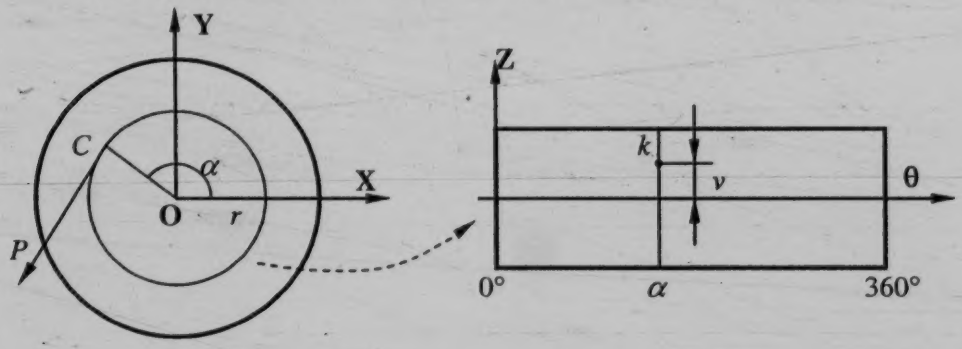
where, $\mathbf{F}_x$, $\mathbf{F}_y$, $\mathbf{F}_z$ and $\mathbf{G}_x$, $\mathbf{G}_y$, $\mathbf{G}_z$ are the x-, y-, and z-component of function **F** and **G**, respectively.

Equation 3-10 shows that when the parameters u and v are fixed, the function

degenerates into the support function of the perspective image. When the parameters s and t are fixed, the function degenerates into the support function of the object image.

Similarly, the spherical light field [70], the uniformly sampled light field [20], and the polar parameterized light field [179] are also special kinds of 4D continuous rayset. The differences among them are the ways the support functions are defined.

A concentric mosaics [166] uses three parameters, radius, rotation angle, and vertical elevation, to parameterize the rays. Therefore, it can be cast as a 3D rayset.



(a) (b)

Figure 3-11: Parameterization scheme used in concentric mosaics.

Assuming that the center of the concentric mosaics is **O**, Figure 3-11(a) shows the projection of the concentric mosaics on the X-Y plane and Figure 3-11(b) shows one of the concentric mosaics, whose radius is r. The parameter (r,α,v) defines a ray, P, that starts from a point C on the circle, where the length of OC is r and the angle between OC and the X axis is α. The direction of ray P is tangent to the concentric mosaic of radius r. The vertical elevation of the projection of ray P on the concentric mosaic is v.

Based on this parameterization method, the support function of the corresponding rayset representation can be defined as:

$$\begin{cases} \mathbf{S}_x(r,\alpha,v) = \mathbf{O}_x + r\cdot\cos(\alpha) \\ \mathbf{S}_y(r,\alpha,v) = \mathbf{O}_y + r\cdot\sin(\alpha) \\ \mathbf{S}_z(r,\alpha,v) \equiv \mathbf{O}_z \\ \mathbf{S}_\theta(r,\alpha,v) = \alpha + \pi/2 \\ \mathbf{S}_\varphi(r,\alpha,v) = \arctan 2(v,f) \end{cases}$$

Equation 3-11: Support function for concentric mosaics.

where, $\mathbf{O}_x$, $\mathbf{O}_y$, and $\mathbf{O}_z$ are the x-, y-, and z-coordinates of the center of the concentric

mosaics, and f is the focal length.

It is observed from Equation 3-11 that when the parameter r is equal to zero, the function degenerates into the support function of the cylindrical panorama. When the parameter α is fixed, the function degenerates into the support function of the semi-parallel image. The term semi-parallel image refers to images, in which different strips are parallel to each other, but within the strips the projection is still perspective [127].

Three novel high-dimensional scene representations, the object centered concentric mosaics, the spiral texture, and the camera field, are presented in Chapter 4 and Chapter 6. Among these approaches, the object centered concentric mosaics is a 3D rayset and the other two are 4D raysets.

3.3 Rayset Rendering

Here different scene reconstruction techniques are reviewed based on the classification given in Subsection 3.1.3. In the next four subsections, the characteristics of approaches in each of the four categories are illustrated: plenoptic sampling, depth warping, texture mapping, and scene morphing. However, the details of individual approaches are not given, as they are already discussed in Section 2.2.

3.3.1 Plenoptic Sampling

Approaches in this category sample the plenoptic function using specially designed parameterization scheme. Since one cannot afford to fully sample the 5D plenoptic function, these approaches reduce the sampling space by either limiting the region of novel views to be generated or making some assumptions about the scene.

Under the rayset taxonomy, approaches in this category can be regarded as techniques that render scenes using raysets only. To sample a scene, all these approaches first define a support function, i.e., the parameterization scheme. They then create a database for storing sampling results of the attribute function. To render a novel view, all these approaches use backward technique. That is, for a given novel ray, these approaches first calculate the parameters of the ray through a reverse process based on the support function. The attributes of the ray are then found using the database of the attribute function.

Approaches in this category can be further classified according to the way that the number of dimensions of the sampling space is reduced. Some approaches use the free space assumption to reduce the number of dimensions to four. This assumption assumes that the space is transparent and that the region of interest is free of occluders. As a result, rays that start from different locations along the direction of the ray are considered the same. Approaches that use this assumption include light field rendering [97], lumigraph rendering [56], spherical light field rendering [70], etc.

Some other approaches reduce the sampling space by sampling only a sub-space of the 5D ray space. For example, the QuickTime VR system [26] samples rays that pass through a pre-defined viewpoint only. Hence, it allows the user to look around, but not walk around in the scene. This reduces the number of dimensions of the sampling space to two. The concentric mosaics approach [166] uses multiple off-centered panoramas to sample rays that start from a planar circular region. It adds one more degree of freedom, and allows the user to move horizontally within the circular region. As a result, the sampling space used has three dimensions.

3.3.2 Depth Warping

Approaches in this category sample both the illumination and the depth information of the scene. They all assume surfaces in the scene to be locally Lambertian. That is, the radiance at any point on a surface is independent of the viewing direction. As a result, with the help of the depth information, a single sample can be used to color novel rays from any direction.

Using the rayset taxonomy, these approaches can be cast as techniques that render scenes using both raysets and per-pixel depth information. To sample a scene, these approaches define a support function, and then store both the color and depth information in the database for the attribute function. To render a scene, these approaches use either the backward or forward technique.

After the depth information is introduced, there is no mapping relation that can be used to calculate the parameters of a given novel ray. Therefore, the direct backward rendering technique used by plenoptic sampling approaches cannot be applied. However, for a given ray, it is possible to search through the rayset to find the corresponding ray.

Here the epipolar constraint is often used to reduce the search space to 1D. Approaches using backward searching include the view synthesis approach proposed by Laveau and Faugeras [89] and the camera field rendering approach presented in Chapter 6.

When the forward technique is used, samples stored in the rayset are enumerated and re-projected to the novel view. To ensure that the closest sample to the novel viewpoint is visible in the generated result, Z-buffer is used to keep the closest samples. Alternatively, one can choose the right rendering order, the so called occlusion compatible ordering, to enumerate the samples so that the one that is farthest away from the novel viewpoint is handled first. An approach that uses Z-buffer is the multiple-center-of-projection image rendering [133]. Approaches that use occlusion compatible ordering include the 3D warping approach [115] and the image-based object approach [125].

Actually, the forward and backward techniques can be combined in the rendering process. The warping-based lumigraph rendering approach [154, 155] and object centered concentric mosaics approach discussed in Section 4.1 are such approaches.

3.3.3 Texture Mapping

Texture mapping approaches have long been used to enhance the realism of rendering results and are considered as the earliest IBR approaches. The most distinct feature of these approaches is the use of 3D geometric proxies to provide coarse shapes of the scene, and the images to supply the details.

Under the rayset taxonomy, these approaches can be considered as techniques that render scenes using raysets that are defined on the geometric proxies of the scenes. To sample a scene, these approaches first define a 3D geometric proxy to approximate it. They then define a support function in the local coordinate of the geometric proxy and use it to sample the photometric and/or geometric details. To render the sampled scene, the 3D geometric proxy is rendered first, followed by rendering the rayset defined on the proxy.

Approaches in this category can be further classified into approaches that can only approximate view-independent photometric details (attribute mapping), approaches that can approximate geometric details (displacement mapping), and approaches that can approximate view-dependent photometric and geometric details (vide-dependent

mapping).

The initial work in the area uses images to sample the view-independent details of the underlying surface only. The details that can be approximated include color [21], specular reflection [15], roughness [13], transparency [49], and normal [14]. Since no geometric details are sampled, the flatness of the geometric proxies become apparent when the viewpoint changes.

Approaches in the second class include different displacement mapping approaches [34, 81, 104, 152] and relief texture mapping [126]. These approaches sample the variation of both color and depth that the underlying surface deviates from the geometric proxy. The parallax and visibility changes can be generated, and therefore, more complex surfaces can be approximated using simple geometric proxies. However, these approaches make no attempt to produce non-diffuse effects, such as highlights and inter-reflections.

Approaches in the last class include view-dependent texture mapping [42], surface light field [186], unstructured lumigraph [18], dynamic texture mapping [30], and the spiral texture mapping presented in Section 4.2. These approaches use multiple images to sample the appearance of an object under different viewing directions. As a result, they can produce view-dependent shading effects.

3.3.4 Scene Morphing

Scene morphing approaches differ from those in the above three categories in that they not only can synthesize novel views but also can generate fluid transition between different scenes.

Under the rayset taxonomy, these approaches can be regarded as techniques that are based on two or more raysets. Before rendering, human interactions are normally required to define corresponding control features for each rayset. During rendering, corresponding control features are interpolated first to provide the in-between control positions, from which the in-between rayset is then generated.

View morphing is the pioneering work of Seitz and Dyer [161] that uses morphing techniques to generate physically correct novel views. This approach interpolates between two conventional images, and therefore, cannot solve the visibility problem. To

address this limitation, an extension, the layer-based morphing, is presented in Section 4.3, which interpolates between two multi-layered images.

3.3.5 Comparison among Categories

In summary, Table 3-1 shows the advantages and disadvantages of approaches in different categories.

Table 3-1: Comparison among different IBR categories

Category	Subcategory	Advantage	Disadvantage	
Plenoptic sampling	Free space	Efficient rendering	Inefficient in sample reuse	Dense samples needed
	Sub-space			Limited novel views
Depth warping	Forward re-projecting	Less samples needed	Lambertian assumption	Need Z-buffer or require single center of projection
	Backward searching			High computational cost
Texture mapping	Attribute	Hardware acceleration	3D proxy needed	No parallax or view-dependent effects
	Displacement			No view-dependent effects
	View-dependent			More samples required
Scene morphing	View	Need a few views only	2D Control needed	Ghosting problem
	Layer-based			More user interactions

Generally speaking, plenoptic sampling approaches render novel views using the closest sampled rays directly. As a result, these approaches are very efficient in terms of computational cost, but not efficient in terms of reusing samples. Some approaches can only provide limited novel views, while others can provide unlimited views, but require a huge number of samples.

Depth warping approaches are much more efficient in reusing the sampled rays. Hence, they normally require much fewer samples than plenoptic sampling approaches. However, these approaches assume that the surfaces in the scene are Lambertian or at least piecewise Lambertian. As a result, they cannot produce view-dependent effects such as highlights and inter-reflections. In addition, these approaches also require accurate depth information, which makes it hard to apply them to certain applications.

Texture mapping approaches make use of 3D geometric proxies, which define the coarse shape for the scene. The geometric proxies help to reduce to number of samples needed. Hardware acceleration is also possible, which is another advantage of these approaches. Their major limitation is the need for human interactions to define the 3D geometric proxies.

Scene morphing approaches also require human interaction to provide 2D control features. It is, therefore, difficult to use as well. However, their advantage is that only a few input images are required and no depth information is needed. Hence, they can be applied to applications such as generating animation for historical sites based on old photos. Scene morphing approaches can also be used to generate special effects, which are out of the scope of IBR.

3.4 Rayset Editing

As discussed before, the plenoptic function can be separated into two mappings. The first mapping defines how to sample the 5D ray space; the second defines what the sampling results are. Such a separation introduces flexibilities. For example, it is possible to alter one of the mappings while keeping the other unchanged. In this section, the effects of editing these two mappings are discussed.

3.4.1 Distortion Filters

A distortion filter is a function, D:$\mathbf{U}\rightarrow\mathbf{U}$ that maps from the parameter space to itself. It changes the first mapping relation only, i.e., it changes the geometry of the rays without altering the information that they carry with. Applying a distortion filter gives warping effects in generated images.

Image warping is a transformation that maps all positions in one image plane to positions in a second plane [52]. Mathematically, a warping is a function that defines a velocity vector for each pixel in the image. As shown in Equation 3-12, appling a warping to an image will warp the pixels in the image to new locations. Therefore, it gives the effect that objects in the image are moved or distorted.

$$\begin{cases} u' = u + \mathbf{F}_x(u,v) \\ v' = v + \mathbf{F}_y(u,v) \end{cases}$$

Equation 3-12: Effect of a warping on an image.

Noticeably, Equation 3-12 is a mapping within the parameter space of an image. As a result, the warping can be cast as an instance of distortion filters, which are defined on perspective images. More generally, the distortion filters for other types of raysets can be defined. Figure 3-12 illustrates the effects of several filters defined for a 4D rayset, the light slab.

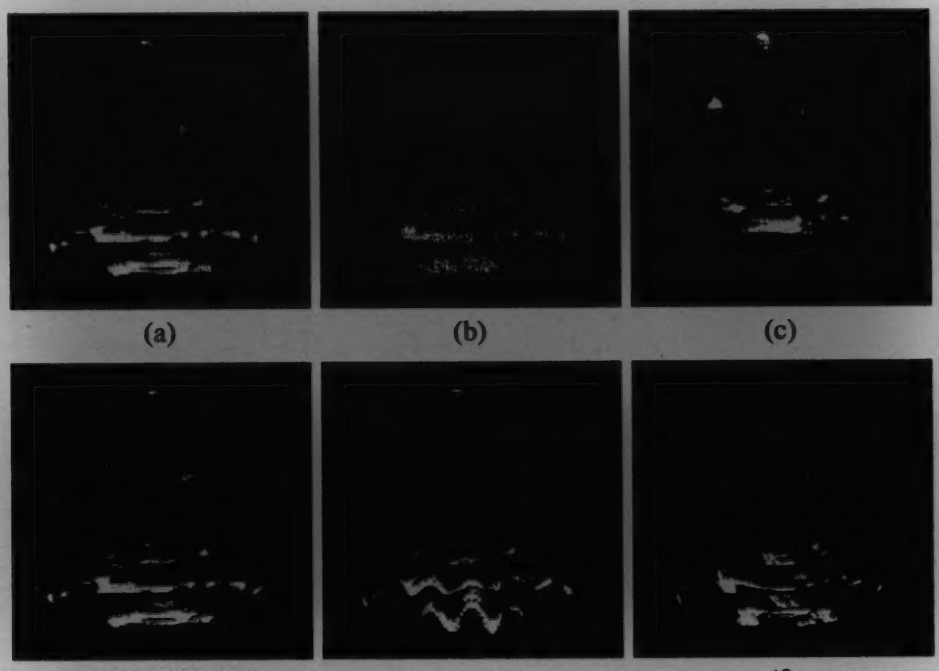


Figure 3-12: Effects of distortion filters on the "Buddha" light field.

Figure 3-12(a) is the rendering result for the Buddha light field without any filter applied. Figure 3-12(b) shows the result of a random filter, which gives the blurry effect. The filter is simply defined by:

$$\begin{cases} u' = u + kU \\ v' = v + kU \\ s' = s + kU \\ t' = t + kU \end{cases}$$

Equation 3-13: Random distortion filter for light slabs.

where U is a random number generator, and k a control parameter.

Figure 3-12(c) shows, what is called, the wide object angle effect. As shown in the image, both the top and the bottom of the Buddha show up in the rendering result. This filter is generated by the following simple filter:

$$\begin{cases} u' = u + ks \\ v' = v + kt \\ s' = s \\ t' = t \end{cases}$$

Equation 3-14: Wide object angle filter for light slabs.

Figure 3-12(d) shows a horizontal ripple effect, which is generated by the following filter:

$$\begin{cases} u' = u \\ v' = v \\ s' = s + k\sin(t/\lambda) \\ t' = t \end{cases}$$

Equation 3-15: Horizontal ripple filter for light slabs.

where λ is the wavelength of the ripple.

Figure 3-12(e) shows a vertical ripple effect, which is generated by Equation 3-16. It is noteworthy that the windy effect is stronger for pixels farther away from the focal plane.

$$\begin{cases} u' = u \\ v' = v + k\sin(t/\lambda) \\ s' = s \\ t' = t \end{cases}$$

Equation 3-16: Vertical ripple filter for light slabs.

Figure 3-12(f) shows a circular ripple effect, which is generated by the following

equation:

$$\begin{cases} u' = u \\ v' = v \\ s' = s + k\sin\left(\sqrt{s^2+t^2}/\lambda\right) \\ t' = t + k\sin\left(\sqrt{s^2+t^2}/\lambda\right) \end{cases}$$

Equation 3-17: Circular ripple filter for light slabs.

The above examples show that using the rayset concept, the idea of image warping can be generalized to higher dimensional cases.

3.4.2 Re-shading Filters

Re-shading filters are filters that change the second mapping relation only. Depending on their properties, they are further classified into global re-shading filters and local re-shading filters.

A global re-shading filter is a function, G:**T**→**T** that maps from the attribute space to itself. Different image processing operations, such as image enhancement and gamma correction, can be classified as global re-shading filters for 2D raysets.

Applying global re-shading filters to high-dimensional raysets can change the appearance of a scene, such as color and brightness. However, global re-shading filters have their limitations. Generally, they do not utilize locality information carried by ray-parameters, and therefore, cannot be used to edit the scene locally.

More interestingly, a local re-shading filter can be defined as a function, L:**U**×**T**→**T** that maps from the product of the parameter and the attribute space to the attribute space. Since the parameter space is used as input of the function, the users can have more controls over the final results.

Recently proposed plenoptic image editing [163] and image-based editing [118] approaches can be cast as applications of local re-shading filters. The first approach allows user to paint or to cut out portion of the scene. The second one makes it possible to relight the scene, to remove unwanted shadow, and to apply new textures on some surfaces, etc.

Chapter 4
Rayset Applications for Static Scenes

In this chapter, three novel IBR approaches that can be used to handle static scenes are discussed. Each approach has different requirements and advantages, and hence, is suitable for certain applications.

The limitation of the existing concentric mosaics approach is that it cannot represent objects and cannot accommodate vertical occlusion. In Section 4.1, a novel object centered concentric mosaics approach is discussed to address these problems. The storage size for an object centered concentric mosaics is about the same as an animation sequence generated with a virtual camera rotated around the object. However, comparing with such an animation sequence, object centered concentric mosaics provides a much richer experience because the user can move back and forth freely in the scene and observe the changes in parallax.

Traditional textures cannot represent the geometric details and view-dependent photometric details of the underlying surface. Displacement maps and view-dependent textures are proposed to solve these problems, but they also have some limitations. In Section 4.2, a spiral texture mapping technique is presented, which can uniformly sample rays that pass through a rectangular texture. Since spiral textures can produce both parallax effects and non-diffuse effects, they can be used to represent shiny complex real objects or environments.

The view morphing approach use one-to-one warping functions to interpolate in-between views. Consequently, when some features are visible in only one of the reference images, the "ghosting" problem will appear. In addition, since the control

features have global effects, there is no way to control different objects in the scene separately. In Section 4.3, a novel layer-based morphing approach is discussed to address these problems. Since the visibility problem is solved, this approach can be used to interpolate novel views for complex scenes.

4.1 Object Centered Concentric Mosaics

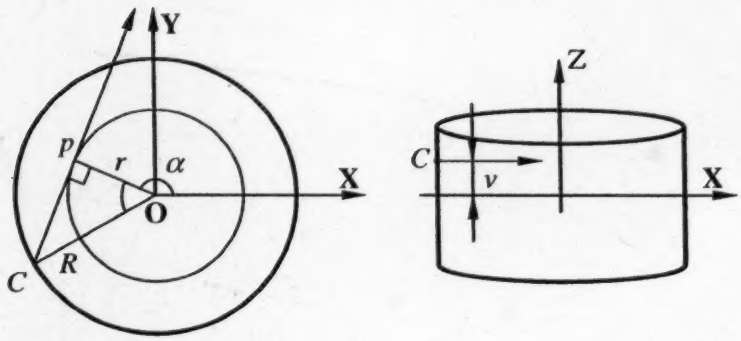
In most cases, when a user walks through a virtual environment, the height of the viewpoint is fixed. Based on this observation, the concentric mosaics approach constrains the camera motion to a horizontal plane and uses manifold mosaics of different radii to depict the scene. It allows a user to move freely in a circular region and to observe lighting changes and horizontal parallax effects. However, vertical distortions still exist. In addition, this approach cannot be used to represent objects.

The rayset taxonomy shows that reversing the sampling direction of a perspective image gives us an object image, which describes the appearance of a point under different viewing direction. Following this idea, the sampling direction of concentric mosaics can be reversed to get, what is called, object centered concentric mosaics, which represents the appearance of an object under different viewing directions.

4.1.1 Parameterization Scheme

Object centered concentric mosaics uses three parameters, radius, rotation angle, and vertical elevation, to parameterize the rays. However, instead of shooting the rays from planar circles on a plane, all the rays start from a bounding cylinder, which surrounds the object tightly. Moreover, the directions of all testing rays are parallel to the X-Y plane.

Figure 4-1(a) shows the projection of the object centered concentric mosaics on the X-Y plane and Figure 4-1(b) shows one of the object centered concentric mosaics. The ray with parameter (r,α,v) is tangent to the circle of radius r. The angle between the X-axis and Op is α, where Op is the line connecting the center of the circle and the tangent point. The vertical elevation, v, determines the height of the starting position of the ray.



(a) (b)

Figure 4-1: Parameterization scheme used in object centered concentric mosaics.

More concisely, this sampling scheme can be defined using the following support function:

$$\begin{cases} \mathbf{S}_x(r,\alpha,v) = \mathbf{O}_x + R\cdot\cos(\alpha+\arccos(r/R)) \\ \mathbf{S}_y(r,\alpha,v) = \mathbf{O}_y + R\cdot\sin(\alpha+\arccos(r/R)) \\ \mathbf{S}_z(r,\alpha,v) = \mathbf{O}_z + v \\ \mathbf{S}_\theta(r,\alpha,v) = \alpha - \pi/2 \\ \mathbf{S}_\phi(r,\alpha,v) \equiv 0 \end{cases}$$

Equation 4-1: Support function for object centered concentric mosaics.

where, $\mathbf{O}_x$, $\mathbf{O}_y$, and $\mathbf{O}_z$ are the x-, y-, and z-coordinates of the geometric center of the bounding cylinder with radius R.

4.1.2 Creation of Object Centered Concentric Mosaics

To create object centered concentric mosaics, the continuous support function defined in Equation 4-1 need to be sampled. In the original concentric mosaics approach, the angular direction is sampled uniformly. But in the radial direction, the radii are sampled at location [0, $1/n^{1/2}$, $2/n^{1/2}$, ... , 1]. Using the same way to sample the object centered concentric mosaics will cause the boundary of the object over sampled and the center part of the object under sampled. Therefore, in object centered concentric mosaics, both the angular direction and the radial direction are uniformly sampled.

The shape of the object to be represented determines the parameters of the bounding cylinder. Here, the ratio between the number of mosaics and the vertical resolution of the mosaic is kept to be the same as the ratio between the diameter and the height of the

bounding cylinder. This constraint helps to achieve similar sampling rate for both the horizontal and the vertical directions. The actual sampling rate needed is determined by the maximum output resolution of the object in novel views.

For each sample ray, both the illumination information and the depth information are sampled. Since the object is surrounded by the bounding cylinder, the depth of intersection must be within the range $[0, (R^2-r^2)^{1/2}]$. Therefore, the depth is quantized using the following equation:

$$d = \begin{cases} 0 & \text{if no intersection exists} \\ 255 - 254 \times \left\lfloor \dfrac{depth}{2\sqrt{R^2 - r^2}} \right\rfloor & \text{otherwise} \end{cases}$$

Equation 4-2: Quantization of depth information.

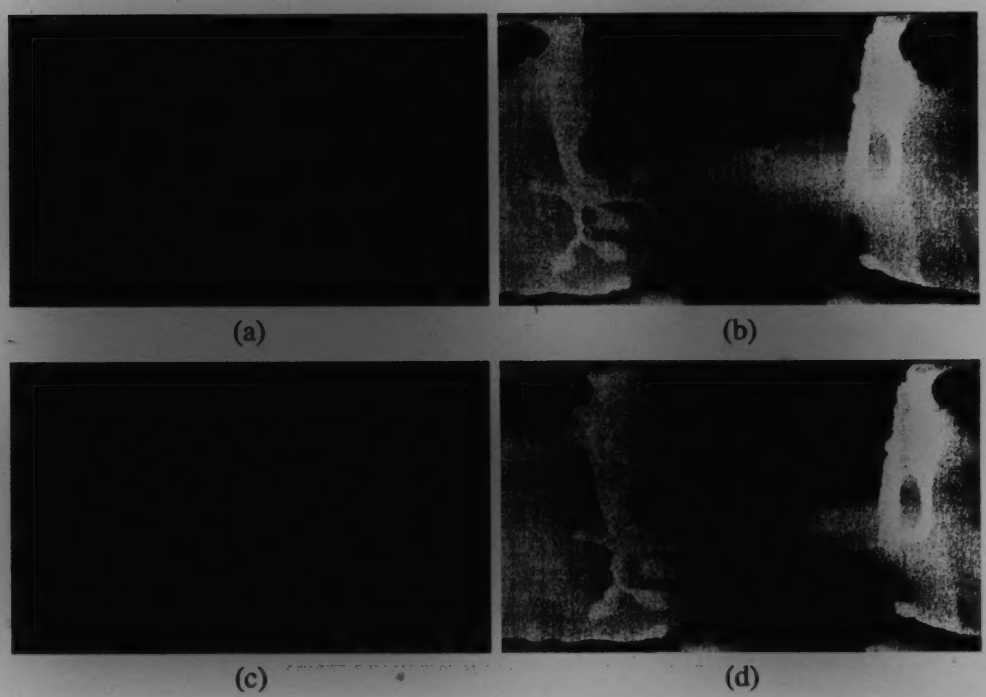


Figure 4-2: Two object centered concentric mosaics for the "Happy Buddha" model.

Figure 4-2 shows two examples of object centered concentric mosaics generated for the synthetic "Happy Buddha" model. Figure 4-2(a) and (b) show the color and depth information for the object centered concentric mosaic of radius zero, respectively. Figure 4-2(c) and (d) depict those for the object centered concentric mosaic whose radius is equal to one third of the bounding cylinder's radius.

To create object centered concentric mosaics for real objects is also relatively simple. Equation 4-1 shows that when the parameter α is fixed, the rayset degenerates into a 2D rayset, T, in which all rays are parallel to each other. Since both parameters r and v are uniformly sampled, the only difference between a sampled rayset T and a discrete parallel image is that in T all the rays start from a cylinder, rather than from a camera plane. Since no intersection should exist outside the bounding cylinder, one can always convert a parallel image into rayset T by adjusting the depth value of each pixel. Therefore, the full object centered concentric mosaics can be converted from a set of depth parallel images, whose horizontal directions evenly sample the angular space.

The above feature leads to a simple way to create object centered concentric mosaics for real objects: Put the object on a turntable and rotate the object along a vertical axis. For each angular direction, sample both the color and depth information using a fixed parallel projected range scanner. The obtained depth parallel images can then be converted into an object centered concentric mosaic. No resampling process is needed since all need to be done is reindex the captured pixels.

4.1.3 Rendering of Object Centered Concentric Mosaics

The rendering process contains three steps, which are the backward mapping in the horizontal direction, the forward mapping in the vertical direction, and then, post-warping. The first two steps generate an intermediate image, whose center of projection is the same as the novel view, but its view direction is adjusted to make it parallel to the X-Y plane. In the intermediate image, different pixels in the same column have the same horizontal viewing direction, i.e. the parameter θ of the corresponding rays are the same. This makes it possible to render one column at a time.

The backward mapping process is similar to the process used in the concentric mosaics approach. Basically, for each column in the intermediate image, the process searches for the corresponding column in a pre-sampled concentric mosaic. This process is very efficient since the two parameters needed, r and α, can be calculated directly using the following set of equations:

$$\begin{cases} r = \sqrt{\|\mathbf{O}' - \mathbf{P}'\|^2 - \dfrac{\|(\mathbf{O}' - \mathbf{P}') \cdot \mathbf{d}'\|^2}{\|\mathbf{d}'\|^2}} \\ \alpha = \arctan 2(\mathbf{d}_y, \mathbf{d}_x) + \pi/2 \end{cases} \quad \text{when } \mathbf{d}_x(\mathbf{O}_y - \mathbf{P}_y) \le \mathbf{d}_y(\mathbf{O}_x - \mathbf{P}_x)$$

$$\begin{cases} r = -\sqrt{\|\mathbf{O}' - \mathbf{P}'\|^2 - \dfrac{\|(\mathbf{O}' - \mathbf{P}') \cdot \mathbf{d}'\|^2}{\|\mathbf{d}'\|^2}} \\ \alpha = \arctan 2(\mathbf{d}_y, \mathbf{d}_x) - \pi/2 \end{cases} \quad \text{otherwise}$$

Equation 4-3: Reverse mapping equations for object centered concentric mosaics.

where, $\mathbf{O}'=[\mathbf{O}_x,\mathbf{O}_y]^T$, $\mathbf{P}'=[\mathbf{P}_x,\mathbf{P}_y]^T$, and $\mathbf{d}'=[\mathbf{d}_x,\mathbf{d}_y]^T$ are the projection of the concentric mosaics center **O**, the start position of ray **P**, and the direction of the ray **d**, respectively.

After the corresponding column is found, forward mapping is applied within the column. This process takes the position of the novel viewpoint and the depth information as inputs, and warps pixels to new locations to eliminate vertical distortion. The warping is conducted in the occlusion compatible order [115]. Therefore, no Z-buffer is needed. This process is also efficient since only 1D warping is involved. Details of this warping process can be found in [126].

Finally, the post-warping process is conducted to apply projective transforms to the intermediate image. The novel view is then produced based on the required viewing direction. The implementation takes advantage of an efficient scanline algorithm [184]. This process can also be performed through texture mapping and can exploit graphics hardware.

(a) (b) (c)

Figure 4-3: Three steps in the rendering process.

Figure 4-3 shows the images generated after each step. As shown in Figure 4-3(a), the

horizontal parallax effect is produced in the backward mapping process. The vertical parallax effect is introduced in the forward mapping process, shown in Figure 4-3(b). Finally the post-warping process adds the foreshortening effect according to the viewing direction, shown in Figure 4-3(c).

4.1.4 Experimental Results

The algorithm is tested using a synthetic model, the "Happy Buddha". The model contains over a million triangles, and therefore, cannot be rendered in real time on current machines. According to the aspect ratio of the bounding cylinder for the model, 175 concentric mosaics are generated, with the vertical resolution set as 336. The angular space is sampled at every 2/3 degrees, i.e. the horizontal resolution of the concentric mosaics is set as 540.

The raw data size is about 120MB. Similar to the concentric mosaics approach and the light field approach, vector quantization [67, 120] can be applied to reduce the size. However, in this experiment, raw images are used directly because it is affordable to load them into memory.

On a Pentium III 733MHz PC with 256MB RAM running Windows 2000, only 0.015 second is needed to generate a frame of size 240×320 (the actual object fills the frame).

(a) (b) (c)

Figure 4-4: Rendering results for different viewing directions.

Figure 4-4 shows the rendering results for different viewing directions. It demonstrates the change in both the highlights and inter-reflections.

(a) (b)

Figure 4-5: Rendering results for different camera settings.

Figure 4-5 illustrates the parallax changes for the same viewing direction but under different camera settings. Figure 4-5(a) is captured using a virtual telephoto lens at a larger distance, while Figure 4-5(b) is captured using a wide-angle lens from a much closer position.

(a) (b)

Figure 4-6: Rendering results for viewpoints with different heights.

The original concentric mosaics approach allows the user to move within a planar circular region only, i.e. the user cannot move up and down. The presented approach gives the user more freedom in changing the height of the viewing position within the top and bottom boundary of the bounding cylinder. Figure 4-6 shows the two rendering results with the same horizontal viewing direction but different heights.

4.1.5 Discussions

An IBR technique that can be used to represent an object is discussed in this section. The

scene representation used, the object centered concentric mosaics, is a 3D rayset. The rendering approach used can be classified as one of the depth warping techniques using a combined forward backward technique. This technique can produce correct parallax effects in both the horizontal and vertical directions, and can also generate horizontal view-dependent shading effects.

The generated object centered concentric mosaics are all multiple-center-of-projection images. However, different from multiple-center-of-projection images, all columns in the mosaics are constrained to be vertical. The constraint makes it possible to find the corresponding column in the concentric mosaics directly. This advantage is very important since one cannot afford to reproject all the columns in different mosaics to render a single image.

Comparing with the image-based object approach, object centered concentric mosaics adds one more dimension. Therefore, non-diffuse effect and self-occlusion are captured. In addition, rendering object centered concentric mosaics uses backward mapping plus 1D warping, which is more efficient than the 3D image warping used in image-based object.

The drawback of the object centered concentric mosaics approach is that it does not capture the appearance of the object in the top and the bottom direction, which makes horizontal surfaces poorly sampled. Therefore, the viewpoint is limited within the height range of the bounding cylinder. However, in many applications, such as to show a new product or to evaluate an architectural model, there is little need to observe the object from the top or the bottom. The object centered concentric mosaics approach is a good choice for these applications since it requires only a reasonable storage and can be rendered in real time.

4.2 Spiral Texture Mapping

Conventional texture mapping uses a single image to represent surface details, and therefore, cannot produce self-occlusion effects and non-diffuse effects. Displacement mapping and relief texture mapping try to simulate self-occlusion by adding depth information. However, they cannot solve all the visibility problems since some surface details may not even be visible from the sampling direction. View-dependent texture

mapping alleviates the problem by combining images taken from different directions. Nevertheless, the sparse set of perspective images used cannot guarantee all surface details are captured, so the problems may still exist.

Here the problem is solved by sampling the texture evenly and densely enough. For a given planar texture, the spiral texture samples all possible viewing directions. In addition, the rays that pass through the texture surface are sampled uniformly.

4.2.1 Uniform Sampling for the Viewing Directions

In order to cover all the possible viewing directions, the range of projection directions should be the whole hemisphere. To provide uniform sampling, previous research [20] subdivides the sphere based on a platonic solid, such as an octahedron or an icosahedron. This approach has the drawback that it cannot use an arbitrarily specified sampling rate. The number of samples increases four times for each subdivision process.

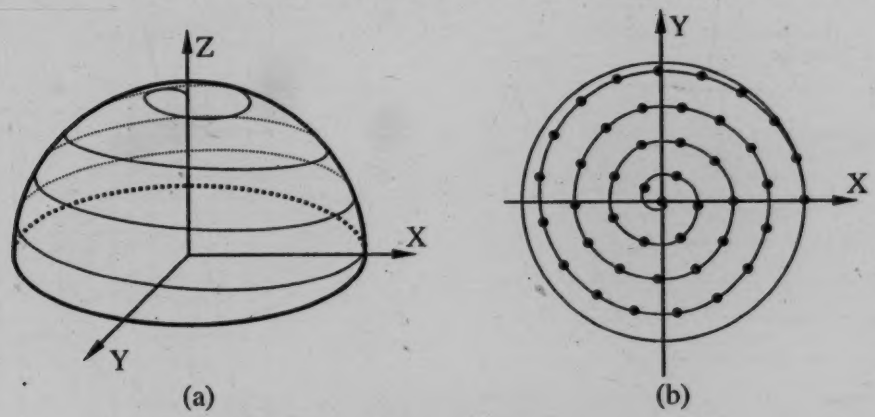


Figure 4-7: Uniform sampling using spiral.

A novel sampling technique, which is based on a spiral, is discussed here. As shown in Figure 4-7(a), a spiral curve is used to sample the hemisphere first. After that, as illustrated using a 2D spiral in Figure 4-7(b), points that are spaced at equal distance along the curve are used to sample the spiral curve itself. The parametric function of the spiral is defined as:

$$\begin{cases} x(\alpha) = \sin(k\alpha)\cos(\alpha) \\ y(\alpha) = \sin(k\alpha)\sin(\alpha) \\ z(\alpha) = \cos(k\alpha) \end{cases}$$

Equation 4-4: Parametric functions of a spiral defined on a sphere.

where, α is the spiral parameter, and k the parameter that controls the density of the spiral curve.

Equation 4-4 defines a spiral that starts from the top of the hemisphere. Any given point on this spiral defines a direction, whose yaw angle is α and pitch angle is $k\alpha$. Clearly, the pitch between adjacent lines of the spiral is $2k\pi$. The length of the spiral can be computed using the following integration:

$$L(\alpha) = \int_0^{\alpha} \sqrt{(x'(\theta))^2 + (y'(\theta))^2 + (z'(\theta))^2}\, d\theta = E\left(k\alpha, -k^{-2}\right)$$

Equation 4-5: Length of the spiral that covers the hemisphere.

where, $E(\phi,m)$ is the elliptic integral of the second kind, which is defined as:

$$E(\phi,m) = \int_0^{\phi} \sqrt{1 - m\sin^2(\theta)}\, d\theta = \int_0^{\sin(\phi)} \frac{\sqrt{1-mt^2}}{\sqrt{1-t^2}}\, dt$$

Equation 4-6: Elliptic integral of the second kind.

To cover the whole hemisphere, the range of the parameter α should be $[0, \pi/(2k)]$. To keep the sampling rate along the spiral the same as that across the spiral, the spiral curve is sampled with a set of points that are spaced at a distance of $2k\pi$. Therefore, the total number of samples needed can be calculated by:

$$n(k) = \left\lfloor \frac{L(\pi/2k)}{2k\pi} \right\rfloor + 1 = \left\lfloor \frac{E\left(\pi/2, -k^{-2}\right)}{2k\pi} \right\rfloor + 1$$

Equation 4-7: Total number of sampling directions needed.

The spiral parameter, α, for any given sample q can be calculated by solving the equation:

$$L(\alpha) = q \times 2k\pi, \text{ where } 0 \le q \le n(k)$$

Equation 4-8: Calculation of the spiral parameter for a given sample.

In practice, after the sampling rate k is decided, the mapping relation between sample

q and the spiral parameter α are pre-computed and tabulated. Normally, the value of k varies from 1/16 to 1/32, and the corresponding numbers of samples are within the range of [42, 164]. Hence, very little memory is required to store the mapping table.

Clearly, when k is small, i.e. the sampling is dense enough, the above sampling scheme gives an equal number of samples per unit solid angle. Different from polyhedron subdivision, the spiral approach can sample the hemisphere using an arbitrary number of samples by adjusting the value of k.

4.2.2 Adaptive Sampling for the Projection Positions

Now, what needs to be considered is how to sample the positions of rays evenly. First, the problem in two dimensions is illustrated. Figure 4-8 shows a ray in both the Cartesian space and the corresponding line space. In the line space, each ray is represented by a point and each set of rays by a region [97]. Assume that one wants to sample the rays that pass through a line segment on the x-axis. It is easy to understand that the corresponding region in the line space for this set of rays is the region enclosed by two sine curves, as shown in Figure 4-8(b). Therefore, under an ideal sampling scheme, the corresponding points of the sampling rays should populate this region uniformly. Obviously, the two-plane parameterization, in which a fixed number of samples is used for different directions, will over sample the areas close to θ=0 and θ=$-\pi$. To avoid this, the number of samples needed is determined according to the length of the segment's projection on different directions.

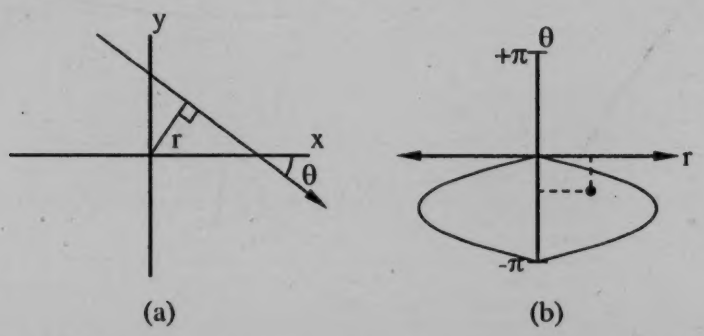


Figure 4-8: A ray under (a) the Cartesian space and (b) the line space.

The extension to three dimensions is straightforward. To uniformly sample the set of rays that pass through a rectangle on the X-Y plane, the number of samples needed is

adjusted according to the area of the rectangle's projection on the plane that is perpendicular to the view direction. To do so, assuming that the texture is sampled using $M \times N$ pixels in the perpendicular direction, for any given view direction (x,y,z), the number of pixels used is $R_x M \times R_y N$, where R_x and R_y are the scaling factors in the x- and y-directions and are defined as:

$$\begin{cases} R_x = \sqrt{\dfrac{z}{\sqrt{z^2 + x^2 y^2}}}\sqrt{1 - x^2} \\ R_y = \sqrt{\dfrac{z}{\sqrt{z^2 + x^2 y^2}}}\sqrt{1 - y^2} \end{cases}$$

Equation 4-9: Scaling factor for adaptively sampling.

The scaling factors are defined to provide the following two properties:

- The number of samples used for a given direction is proportion to the projection size of the rectangle along the direction, i.e.:

$$R_x \times R_y = \frac{z}{\sqrt{z^2 + x^2 y^2}}\sqrt{1 - x^2}\sqrt{1 - y^2} = z$$

- The number of samples along the x- or y-direction does not change if the projection is perpendicular to the x- or y-axis, i.e.:

$$R_x = 1 \Leftrightarrow x = 0 \quad R_y = 1 \Leftrightarrow y = 0$$

In summary, the sampling scheme for spiral texture can be defined by the following support function.

$$\begin{cases} \mathbf{S}_x(u,v,l) = u / R_x \\ \mathbf{S}_y(u,v,l) = v / R_y \\ \mathbf{S}_z(u,v,l) = 0 \\ \mathbf{S}_\theta(u,v,l) = L^{-1}(l) \bmod 2\pi \\ \mathbf{S}_\phi(u,v,l) = kL^{-1}(l) \end{cases}$$

Equation 4-10: Support function for spiral textures.

4.2.3 Comparison with the Two-plane Parameterization

Here the numbers of samples needed for the spiral texture parameterization and for the two-plane parameterization under the same sampling density requirement are compared.

Again, it is assumed that the texture is sampled using $M \times N$ pixels in the perpendicular direction. For the spiral texture parameterization, the total number of samples required can be computed by:

$$m = \frac{\int_0^{\pi/2k} M \times N \times z(\theta) \times \sqrt{(x'(\theta))^2 + (y'(\theta))^2 + (z'(\theta))^2}\, d\theta}{2k\pi}$$
$$= \frac{M \times N\sqrt{1+k^2}}{4k^2\pi} + \frac{M \times N}{4\pi} \operatorname{arctanh}\left(\frac{1}{\sqrt{1+k^2}}\right)$$

Equation 4-11: Number of samples needed for spiral texture parameterization.

For the two-plane parameterization, in order to use a single light slab to cover the whole hemisphere[1], one of the defining planes has to be put at infinity. Under the same sampling density requirement, i.e. the directional space is sampled at $2k\pi$ interval and the maximum number of samples needed per direction is $M \times N$, the total number of samples required can be computed by:

$$m' = \frac{\pi}{2k\pi} \times \frac{\pi}{2k\pi} \times M \times N = \frac{M \times N}{4k^2}$$

Equation 4-12: Number of samples needed for two-plane parameterization.

The ratio between the numbers of samples needed by these two parameterization schemes can be calculated as:

$$h = \frac{m'}{m} = \frac{\pi}{\sqrt{1+k^2} + k^2 \operatorname{arctanh}\left(\frac{1}{\sqrt{1+k^2}}\right)}$$

Equation 4-13: Ratio of the numbers of samples needed.

Equation 4-13 shows that when $k \to 0$, the ratio $h \to \pi$. That is, when the hemisphere is sampled dense enough, the number of samples needed under the new parameterization scheme is $1/\pi$ of the number of samples needed under the two-plane parameterization scheme. In practice, when k is within the range of [1/16, 1/32], the corresponding value of h is within the range of [3.09, 3.13].

---

[1] Use more than one light slab to cover the hemisphere need even more samples.

4.2.4 Creation of Spiral Textures

To generate spiral textures, the underlying object is sampled. Here, how to create spiral textures for real objects is discussed. Generating spiral textures for synthetic scenes is similar and more straightforward.

Similar to the surface light field approach, the geometric model and some photos of the object are required. The geometric model can be obtained from a range scanner. The photos should sample the object from different angles and the camera should be calibrated at each location. Depending on the application, the user can decide on how many spiral textures to approximate the object and where to place them. A support rectangle is then specified for each texture using the coordinates of the four corners of the rectangle. The support rectangle of the texture serves as the image plane for both the view-independent component and the view-dependent component. The user can also specify the resolution, a.k.a. sampling rate, for each texture.

The view-independent part of a spiral texture can be considered as a layered depth image. Depending on the application, the user can choose either a parallel projection or a perspective projection. If the latter one is picked, the center of projection should be supplied by the user.

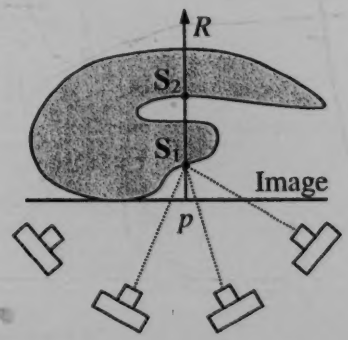


Figure 4-9: Calculation of the view-independent component.

Here the problem is to determine both the depth information and the diffuse color information for each pixel of the layered depth image. As shown in Figure 4-9, the depth information for a given pixel p can be determined by sending out its corresponding ray R from the image plane and by finding the intersections between R and the geometric model of the object. During intersection calculations, only surfaces that are facing towards the image plane are considered. Distances between these intersections and the image plane

are computed and quantized according to the maximum depth available. The quantized values are then stored in the layered depth images in ascending order.

To find the diffuse color information of each intersection requires more computations. The idea here is to find out what the colors of the intersection are in different photos, and then to extract the diffuse color portion from these colors. To do so, rays are sent from the intersection, **S**, to viewpoints of all photos available to test for visibility of **S** in these photos. The color in the photo where **S** is visible is then used as a color sample of the intersection of **S**.

Different strategies can be used to extract the diffuse color. Simple histogram-based operations, such as the mean, median, or mode, can be used. More complex approaches for estimating diffuse albedo [189] can also be applied. Histogram-based operations are used in the current implementation. In particular, median is used in the experiment since it is insensitive to the presence of outliers.

The view-dependent part of the spiral texture can be considered as a set of images with different resolutions. No matter which kind of projection is used in the view-independent part, parallel projection is always used in view-dependent part.

According to the sampling rate requirement, the number of images needed is determined using Equation 4-7. To generate any one of these images, the corresponding spiral parameter α is determined using the lookup table, which is pre-generated by solving Equation 4-8. The direction of projection, **d**, is then calculated using Equation 4-4. Now, the scaling factors according to **d** are computed using Equation 4-9. Finally, multiplying the scaling factors and the user-specified resolution together gives the required resolution to sample the image plane for this view direction.

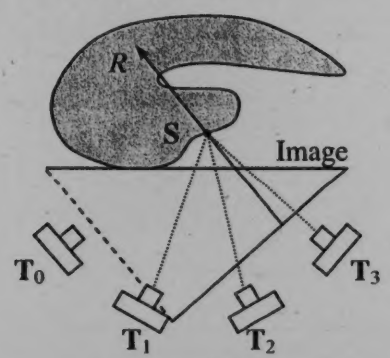


Figure 4-10: Calculation of the view-dependent component.

Now, to calculate the non-diffuse color information for each pixel in the image, it is required to find out the color of the pixel first, and then subtract from it the diffuse portion. To do so, for any given pixel in the image, a ray R is emitted from the image plane along the direction of projection $\mathbf{d}$ to intersect with the object. Different from the view-independent component, here only the closest intersection between R and the geometric model of the object is required.

If such an intersection, $\mathbf{S}$, is found, the color it should have is determined from available photos. The k-nearest neighbor interpolation approach [18] is adopted here. Assume that the intersection is visible in photo i, whose position is $\mathbf{T}_i$. The dot product between $\mathbf{d}$ and $(\mathbf{T}_i-\mathbf{S})$ is computed. The four photos that give the highest dot product values are found, and then three of them are used for interpolation. Assume that the fourth photo has the smallest dot product value, which is w, the weights for each of the first three photos are calculated by $\mathbf{d}\cdot(\mathbf{T}_i-\mathbf{S})-w$. The weighting technique guarantees a smooth blending since the weight of a photo will drop to zero before it is no longer used for interpolation.

The color obtained from interpolation tells us what the intersection $\mathbf{S}$ looks like from direction $\mathbf{d}$. The diffuse color portion of intersection $\mathbf{S}$ can be found by simply projecting $\mathbf{S}$ to the layered depth image computed above. If multiple textures are used to sample the object from different directions, which layered depth image is used depends on the normal of the surface at location $\mathbf{S}$.

Now the diffuse color portion can be subtracted from the color of the intersection so that only the non-diffuse portion is left. Since the color of the intersection viewed from the current direction may have lower intensity than the diffuse color, the difference can be a negative value. To sample the difference using eight bits, a color for zero difference is defined globally. This color is added to the color difference and the result is clamped to within the range of [0, 255]. Which color should be used for the zero difference value depends on the average brightness of the object's diffuse colors.

4.2.5 Rendering of Spiral Textures

The above creation process generates an intermediate representation of the object, and the geometric model and photos of the object are therefore no longer required. The rendering

process for spiral textures can also be separated into two steps: rendering the view-independent component and rendering the view-dependent component. This is a nice feature since it makes it possible to improve the rendering result progressively. That is, the rendering system can choose to perform only the first step when a high interactive rate is required, and complete both steps when a high quality rendering result is needed.

Rendering the view-independent component is similar to the problem of rendering layered depth images. Since multiple textures can be used to define the same object, the rendering algorithm should be able to find the closest point from different layered depth images. This brings an extra problem to forward image warping approaches since the occlusion compatible order [115] does not exist among images taken at different centers of projection. In image-based object [125], the problem is solved by requiring the six layered depth images to share the same center of projection. This is a good idea, except it may not work well when more than six planes are used to approximate the underlying object more closely. Using a common center of projection for all these surfaces may introduce large distortions.

To avoid the above problems, a backward searching approach is applied here, which is similar to the image-based ray tracing approach [31, 101].

As shown in Figure 4-14(b), the rendering results of the view-independent component gives us the correct projection of the object viewed from a given viewpoint. For Lambertian surfaces, this is already good enough. For shiny surfaces, the view-dependent component is rendered to add highlights and inter-reflections.

Rendering view-dependent component is basically an interpolation process. For a given ray, the closest rays are sampled and interpolated. To find the closest samples depends on the sampling scheme used. In two-plane parameterization, since rays are not uniformly sampled, the closest samples can be found easily by intersecting the ray with the two planes. When the directions are sampled using the polyhedron subdivision approach [20], searching for the closest samples involves calculating the intersection between the ray and the polyhedron, followed by a recursive subdivision process to locate the patch. When the unstructured sampling scheme is used [41], even more computations are needed. Basically, the given ray is compared with the corresponding rays in all the photos available.

In the new approach, using the property of the spiral parameterization scheme, the closest available samples can be found using a direct and efficient procedure. The procedure can be best explained as a two-step process. Considering the view-dependent component as a set of parallel projected images, images that sample the texture from the closest directions are determined. Then the pixel that should be used in each of these images is located.

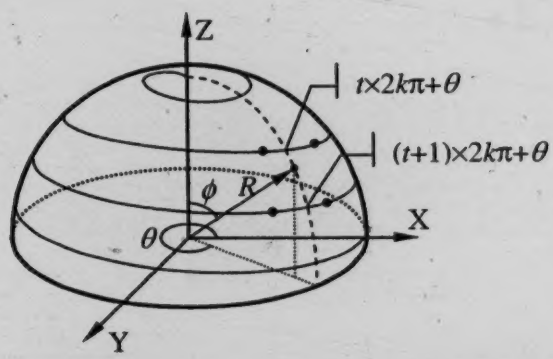


Figure 4-11: The closest directions sampled.

As shown in Figure 4-11, the first step starts from converting the given ray R, defined in the texture's local coordinate using a vector, into two angles, i.e., the yaw angle θ and pitch angle ϕ. Then, the number of rounds the spiral has rotated on the hemisphere before it reaches this direction is estimated using the following expression:

$$t = \left\lfloor \frac{\phi - \theta \cdot k}{2k\pi} \right\rfloor$$

Equation 4-14: The number of rounds for spiral curve.

Four closest samples are used in the interpolation. As shown in Figure 4-11, two of the samples lie on the t^{th} round of the spiral curve, and the other two samples lie on the $t+1^{th}$ round of the curve. Obviously, the spiral parameter α for the first two samples should be close to and on the two sides of value $t\times 2k\pi+\theta$. The other two samples should have α values close to and on the two sides of the value $(t+1)\times 2k\pi+\theta$. Therefore, the lookup table can be used to find images that sample these directions.

In the second step, pixels that should be used in these images are located. Instead of determining the pixel directly using the parameter of the given ray [20, 97], the depth correction [56] is applied to accommodate for the amount that the real object deviates

from the approximate rectangle.

4.2.6 Experimental Results

The "fish" data used in surface light field is used here for experiments [186]. This dataset contains a geometric model and 638 useable photos of a small ceramic fish. The geometric model of the fish, obtained by a range scanner, contains 130K triangles.

In the experiment, five spiral textures are generated for the five sides of the fish, excluding the bottom. According to the aspect ratio of the fish, 270×310 pixels are used to sample the front and back, 183×310 for the left and right, and 270×183 for the top. These resolutions are actually higher than the size of the fish shown in the original photos to avoid the aliasing problem.

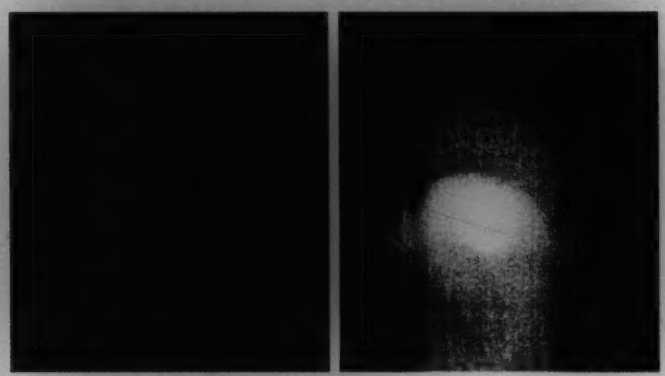


Figure 4-12: View-independent component for the "Fish" dataset.

A parallel projected layered depth image is used for the view-independent component of each texture. Two layers are used for the left and right sides, and one layer is used for the front, back, and top sides. The depth information is quantized using eight bits. The total size for the view-independent components of the five textures is 1.3MB before compression, and 460KB after LZW compression. Figure 4-12 shows the diffuse color information and depth information for the front side texture.

Two different sampling rates are experimented for the view-dependent component. For a low sampling rate, the value of parameter k is equal to 1/16, which yields 42 sample directions for the hemisphere. A higher sampling rate with k equals to 1/24 yields 93 samples. This means that the unit sphere is sampled using 84 and 186 directions, respectively. This is much lower than that is used in uniformly sampled light fields [20],

where 64K and 245K samples are used to sample the unit sphere. It is also comparable with that is used in Lischinski and Rappoport's approach [101], where 66, 258, and 1026 directions are used.

Before compression, for the view-dependent components, the total size of the five textures is 20MB with the low sampling rate and 44MB with the high sampling rate. Since only non-diffuse portions exist, these images are not as colorful as the original photos. Hence, color compression can be applied to compress the color space from true color to 256 colors. Color compression brings the size down to 6.7MB for the low sampling, and 14MB for high sampling rate, without noticeable quality degradation (see Figure 4-13). Using 12-dimensional vector quantization can also further compress the data. For each texture, a codebook is generated with 256 codewords, each representing the colors of four pixels. This approach compresses the data further to 1.7MB and 3.5MB, respectively.

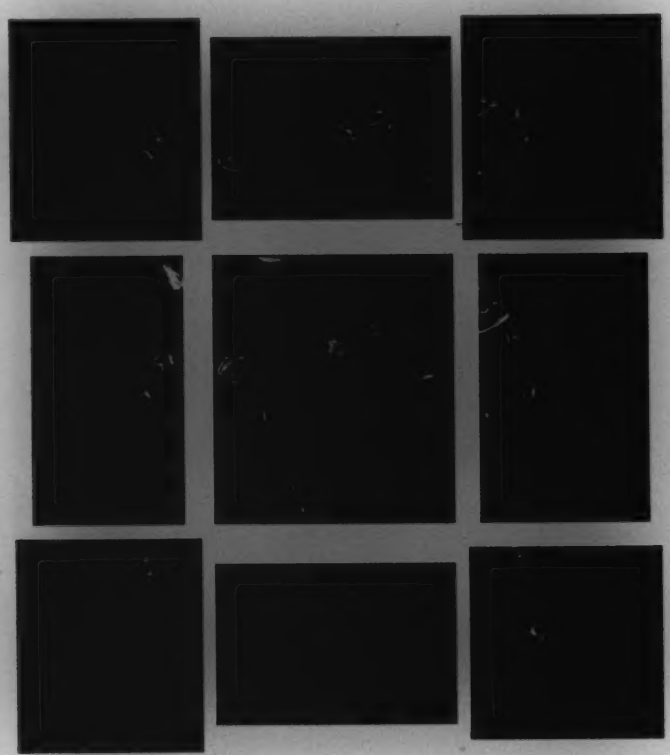


Figure 4-13: Samples of the view-dependent component (after color compression).

Figure 4-13 shows several samples of the view-dependent component[1]. It shows that different samples have different resolutions. It is noteworthy that due to the adaptive sampling, the fish shown in these photos are not as distorted as the Buddha shown in the light field rendering approach [97], which is sampled using the two-plane parameterization scheme.

(a) (b) (c) (d)

Figure 4-14: Rendering process (a)(b) view-independent (c)(d) view-dependent.

Figure 4-14 shows the decomposed rendering process. Figure 4-14(a) is generated using the view-independent component of the left texture only. Figure 4-14(b) uses both the left and front textures, which gives the correct projection of the fish from this view direction. The view-dependent component of the left texture is added in Figure 4-14(c), in which only half of the fish has highlights. The view-dependent part of the front side is added in Figure 4-14(d). One can hardly observe any seam from the image, even though the highlights are obtained from two different textures. However, the seams can be observed if an animation generated is carefully examined. To eliminate the seams through interpolating between different textures will be studied in the future.

To render the fish with output resolution of 240×320 pixels, the current implementation takes about 0.3 second on a 1.4GHz Pentium 4 PC with 256MB RAM running Windows 2000. About 0.1 second is used to render the view-independent component and 0.2 second for the view-dependent component.

---

[1] For illustration purpose only, blue color is used here to indicate no intersections, instead of the color for zero difference used.

4.2.7 Results Evaluation

The results generated are compared with the original photos. Three photos, which have rich highlight information, are selected. The virtual camera parameter is set to be the same as those used in these photos to generate the rendering results. It is noteworthy that the rendering process uses the resampled view-dependent information, and therefore, does not utilize the three photos directly.

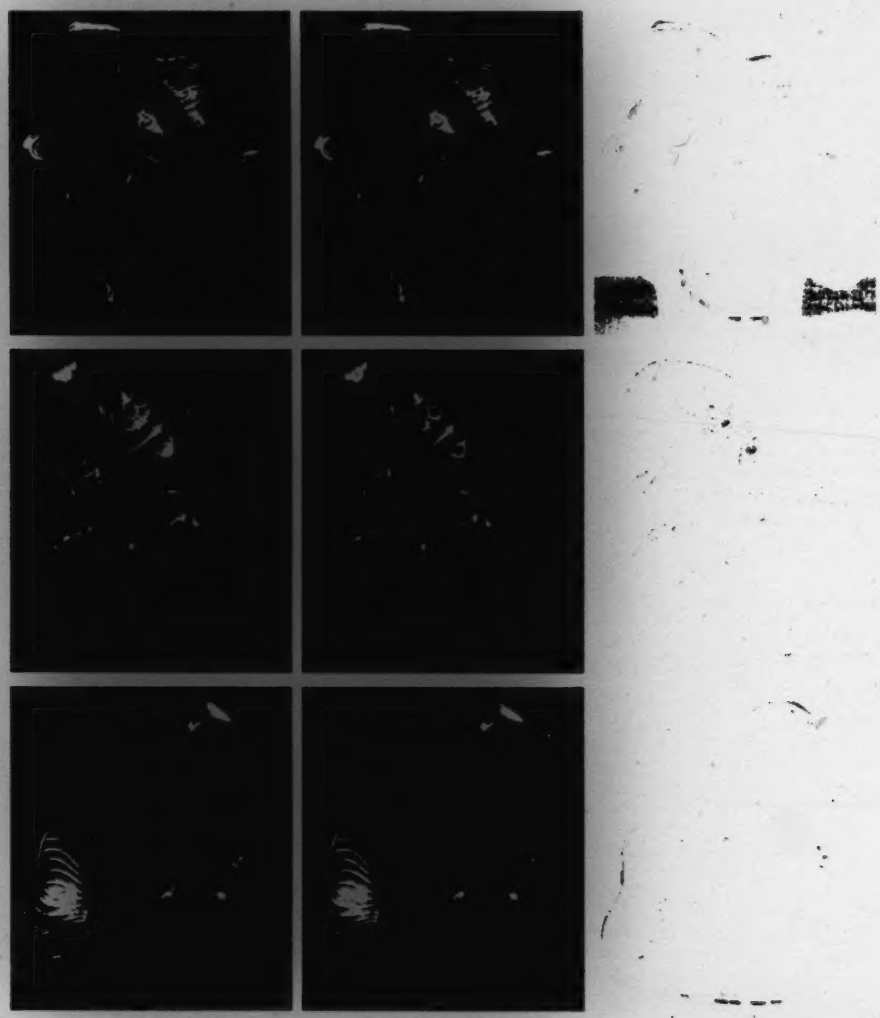
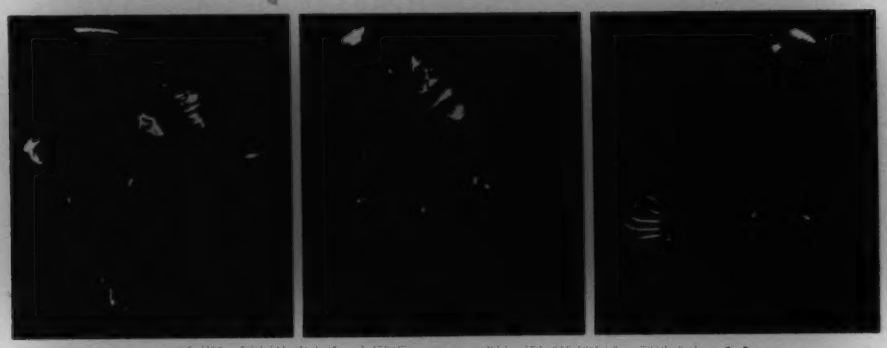


Figure 4-15: Rendering results evaluation for different views.

Figure 4-15 shows the comparison results. Images in the first column are the original photos. Those in the second column are the rendering results using the high sampling rate. The third column shows the differences between images in the first two rows. The

darker the color, the larger the error is. The results show that the spiral texture mapping technique can faithfully reproduce the appearances of the shiny fish in all directions.



(a) (b) (c)

Figure 4-16: Rendering results using the low sampling rate data.

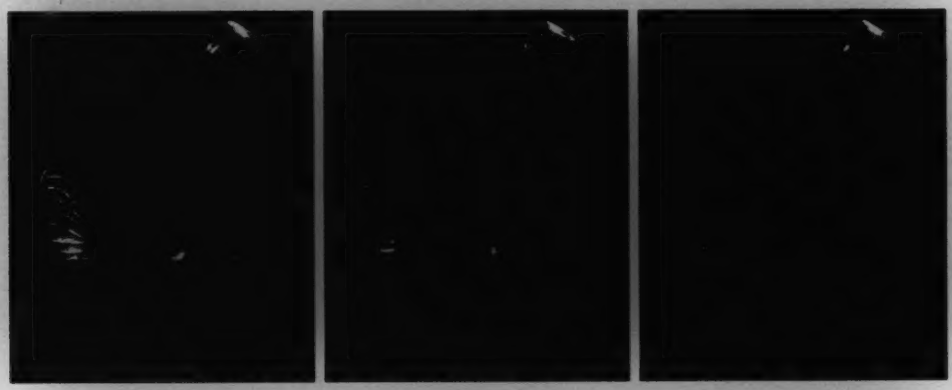
Figure 4-16 shows the rendering results using the low sampling rate version. It shows that dropping half of the view-dependent samples does not significantly degrade the quality of rendering results. Careful examination on Figure 4-16(c) indicates that some highlights on the tail of the fish are lost from this particular view direction. However, it is interesting to see that highlights of the golden stripes on the tail are preserved. A possible explanation is that the highlight for the ceramic has a higher frequency than that for the golden stripes. Therefore, the highlights for the ceramic are not captured in the low sampling rate version, while the highlights for the golden stripes are.

(a) (b) (c) (d)

Figure 4-17: Rendering results under different compression approaches.

When compressions are applied to the view-dependent component, only highlights and inter-reflections of the object are affected. Figure 4-17 compares the results with different compressions schemes using an area with both highlights and inter-reflections (on the body of the fish). Figure 4-17(a) is the original photo. Figure 4-17(b) is the result

generated using uncompressed view-dependent component, which faithfully reproduces details of the original photo, even though they are not as sharp. Color compression is used in Figure 4-17(c), in which the result looks identical to the uncompressed version. In Figure 4-17(d), the view-dependent component is compressed using 12-dimensional vector quantization. The result shows that the quality of the image does not degrade too much.



(a) (b) (c)

Figure 4-18: Results rendered by the surface light field approach.

For comparison, Figure 4-18 shows the results generated by the surface light field approach [186]. The corresponding camera parameters are unknown and therefore may not be the same as the one used in Figure 4-15. Comparison shows that rendering results using uncompressed high sampling rate spiral textures are at least comparable with results using uncompressed pointwise faired surface light field, shown in Figure 4-18(a). However, the former one requires 46MB (included view-independent part) in size and the latter one 177MB. This difference is mainly caused by the uniform sampling scheme used. The spiral textures compressed using vector quantization (3.5MB + view-independent part) have comparable size with the surface light field compressed by principal component analysis (2.5MB + size of the geometry model), shown in Figure 4-18(b). However, the new approach preserves the highlights on the tail and inter-reflections on the body of the fish better. It also eliminates artifacts around the left fin of the fish. Since a less aggressive compression is used, the result is also much better than that of the surface light field compressed by vector quantization (8.1MB + size of the geometry model), shown in Figure 4-18(c).

4.2.8 Discussions

A new IBR technique, the spiral texture mapping, is presented in this section. This approach combines the ideas of using depth information to solve visibility problems (displacement texture mapping) and multiple samples to produce non-diffuse effects (view-dependent texture mapping). The view-independent information is separated from the view-dependent information. As a result, the visibility problems are solved solely based on the depth information in the view-independent component, while non-diffuse effects are produced through interpolating the view-dependent component.

Comparing to existing image warping based approaches [125, 126, 152], the main improvement is in adding another view-dependent component. Therefore, shiny objects or environments can be represented. When using multiple spiral textures to fully represent an object, the view-independent component is similar to the image-based object since both approaches use multiple layered depth images. However, there are subtle differences between the two: (1) the use of parallel projection is allowed, and therefore, a user can use a more complex shape to closely approximate the object; and (2) the extracted diffuse colors are stored rather than the colors that are observed from a single direction.

Although the approach does sample appearances of a scene or object from different directions, it requires much less samples than light field, lumigraph, or related approaches [20, 56, 97]. This is because (1) under the same sampling rate requirement, the sampling scheme needs less than one third of the samples needed by the two-plane parameterization scheme; (2) the use of depth information and the separation of view-independent and view-independent information make it possible to generate comparable results with a lower sampling rate.

The new sampling scheme, which is defined on a spiral, can uniformly sample both directions and positions of rays that pass through a planar rectangle texture. Therefore, one can always use multiple rectangles to approximate complex objects and samples these rectangles. For some objects, using six or more rectangles to enclose them gives a much tighter bound than using spheres. As a result, fewer useless samples exist in the spiral texture sampling scheme than those in the two-sphere parameterization or sphere-plane parameterization.

It is noteworthy that using multiple textures to sample an object does not introduce redundancy, as long as the corresponding support rectangles form a convex solid. This is because even though the two adjacent textures may sample the same direction, the projected positions they sample do not overlap. As a result, using multiple textures may actually decrease the number of samples needed if such a setting can provide a tighter bound for the object.

Besides reducing the number of samples needed, adaptively sample the projection position also helps to reduce the aliasing effect. The traditional fixed sampling scheme over-samples the area when projection direction is close 0 and π. This introduces high frequency information, which will cause aliasing in rendering results.

4.3 Layer-based Morphing

View morphing approach can generate physically correct in-between views based on two input images. However, when some features are visible in only one of the reference images, "ghosting" problems will appear in some in-between images. In addition, since the control features have global effects, there is no way to control different objects in the scene separately.

These two problems can be solved through a novel layer-based morphing approach discussed in this section. Layer-based morphing applies multi-layered images into the morphing area. It treats the cases when physically correct novel views between two input images are needed and no 3D model, depth, or disparity information is available.

4.3.1 Warping Based on Regions

First, different regions are used to control different objects or different parts of an object so that they can be morphed separately. Given two reference images I_0 and I_1, user interaction is needed to indicate corresponding regions in both images. Generally, a region R can be defined as the interior of a closed 2D geometric shape. Polygons, which can have holes, are used in this approach.

Different regions can overlap each other. Each region has its own priority. A region has a higher priority than all the regions that it covers. Each region has associated control features, which have local effects inside the region only.

Before computing the intermediate frame I, the locations of all the regions R_m (m=0...M) in this frame are calculated. This can be done by interpolating the positions of R_m^0 and R_m^1, where R_m^0 and R_m^1 are the corresponding regions in I_0 and I_1, respectively. Linear interpolation works well for static objects and is easy to use. However, it may introduce distortions when interpolating complex motions. Sederberg and Greenwood [158], Sederberg et al. [157] discuss how to smoothly blend between two 2D polygonal shapes. These approaches can all be applied under the layer-based morphing framework.

After the interpolation of corresponding regions, the outlines of regions of the in-between frame are obtained. It is possible that these regions overlap each other. When a particular point **h** is inside more than one region, the one that has the highest priority is used to compute the color of **h**. To determine which region is visible point by point is time consuming. A scanline algorithm is applied since it can employ spatial coherence.

After **h** is determined to belong to region R_m in frame I, it is warped toward the corresponding regions R_m^0 and R_m^1 in images I_0 and I_1, respectively. Only the control features associated with region R_m are used to control the warping.

The warping function should guarantee that points inside the region will remain inside the corresponding region after warping. Assuming that W_0 and W_1 are the warping functions that map from I to I_0 and I_1, respectively, then the following condition needs to be satisfied:

$$\mathbf{h} \in R_m \Leftrightarrow W_0(\mathbf{h}) \in R_m^0 \Leftrightarrow W_1(\mathbf{h}) \in R_m^1$$

Equation 4-15: Constraint on the warping function used.

In this approach, the mapping method proposed by [11] is used. To satisfy the above constraint, all the segments on the boundary polygon of the region serve as control segments by default.

4.3.2 Construction of Multi-layered Images

Morphing by regions gives the flexibility to control different objects in the scene separately. However, holes will appear when the covering regions are moved away. To solve this problem, the color information that is not visible from the viewpoints of the reference images is used.

Recovering the hidden parts for each region is time-consuming and unnecessary.

Here, different regions are assigned to several layers. The bottom layer L_N corresponds to the background of the scene. Each of the upper layers L_n (n=0…N–1) corresponds to some foreground objects in the scene. A layer L_n has an associated support area A_n, which is the union of all the regions that belong to this layer:

$$A_n = \bigcup_{i=p}^{q} R_i, \left(R_p, R_{p+1}, \cdots, R_q \in L_n\right)$$

Equation 4-16: Support area of a layer.

Therefore, in order to solve the visibility problems, the unknown (hidden) part inside the support area of each layer is recovered. If the environment is controllable, one can take photos or generate synthetic images for all the layers and composite them together to get a multi-layered image. However, sometimes it may be difficult, if not impossible, to obtain these images directly. Here, a multi-layered image is created from a single image source automatically based on existing region information.

Several papers describe techniques to recover hidden parts of an image. For example, the hierarchical polynomial fit filter, which is proposed by Burt [19], can be used to recover the missing information using a single image. Debevec et al. [42] also propose an object-space hole-filling method, which requires 3D models of objects.

The hierarchical polynomial fit filter is employed here. To apply the polynomial fit filter, a support image S is required. The resolution of S should be the same as that of the reference image. The value of '1' in a given position of the support image means that the color of the source image at this position is known and the value of '0' otherwise. The goal is to recover the unknown (hidden) part using the known (visible) part.

To get support image S_n for a given layer L_n, a binary image T_n is created so that for a given pixel **h**:

$$T_n(\mathbf{h}) = \begin{cases} 1 & \text{if } \mathbf{h} \in A_n \wedge \mathbf{h} \notin \bigcup_{i=0}^{n-1} A_i \\ 0 & \text{otherwise} \end{cases}$$

Equation 4-17: Binary image generated based on the support area.

Image T_n actually indicates the visible area of layer L_n. To offset the aliasing effect on boundaries of layers, the confidence values of pixels near the boundaries are reduced. Hence, the above image T_n is eroded by a Gaussian filter to get the support image S_n:

$$S_n(\mathbf{h}) = \begin{cases} 0 & \text{if } T_n(\mathbf{h}) = 0 \\ \sum_{u,v} \text{Gaussian}(u,v) \cdot T_n(\mathbf{h}_x + u, \mathbf{h}_y + v) & \text{if } T_n(\mathbf{h}) = 1 \end{cases}$$

Equation 4-18: Support image used by the hierarchical polynomial fit filter.

Based on the support image S_n and the source image I, the hierarchical polynomial fit filter is employed to generate the layer image I_n. Different layer images are then composited to form the multi-layered image.

4.3.3 Morphing between Multi-layered Images

The multi-layered images generated are then used in the morphing process. For a given point **h** in the in-between view, the region it belongs to is determined. Assume **h** belongs to region R_m, which itself belongs to layer L_n, it can be warped toward the corresponding regions R_m^0 and R_m^1, and I_n^0 and I_n^1 are used to compute the color using the following equation:

$$I(\mathbf{h}) = I_n^0(W_0(\mathbf{h})) + \left(I_n^1(W_1(\mathbf{h})) - I_n^0(W_0(\mathbf{h}))\right) \times t \quad t \in [0,1]$$

Equation 4-19: Interpolation of the corresponding pixels in layer images.

where, W_0 and W_1 are the warping functions that map from I to I_0 and I_1, respectively.

```
for each region j
  Calculate position of Rj using Rj^0 and Rj^1;
for each pixel h in the in-between image {
  Determine the region Rm to which h belongs;
  Determine the layer Ln to which Rm belongs;
  Calculate the location W0(h) using the features
defined for regions Rm^0 and Rm;
  Calculate the location W1(h) using the features
defined for regions Rm^1 and Rm;
  Interpolate the colors between the two layer images,
In^0 and In^1, of layer Ln;
}
```

Figure 4-19: Layer-based morphing algorithm.

Although R_m^0 may be invisible in I_0 and R_m^1 may be invisible in I_1, they should be visible in I_n^0 and I_n^1 since regions that cover R_m^0 or R_m^1 should be separated into the upper layer. The visibility problems can be solved now since the missing information in I_n^0 and I_n^1 is recovered already. The overall morphing algorithm is described more precisely in Figure 4-19.

4.3.4 Experimental Results

Since the visibility problems are solved, layer-based morphing can also be used to interpolate complex scenes. Here an architectural model is used as an example. Figure 4-20(a) and (b) shows two rendered images. It is noticed that visibility relations are changed in many locations such as the roof and the street light.

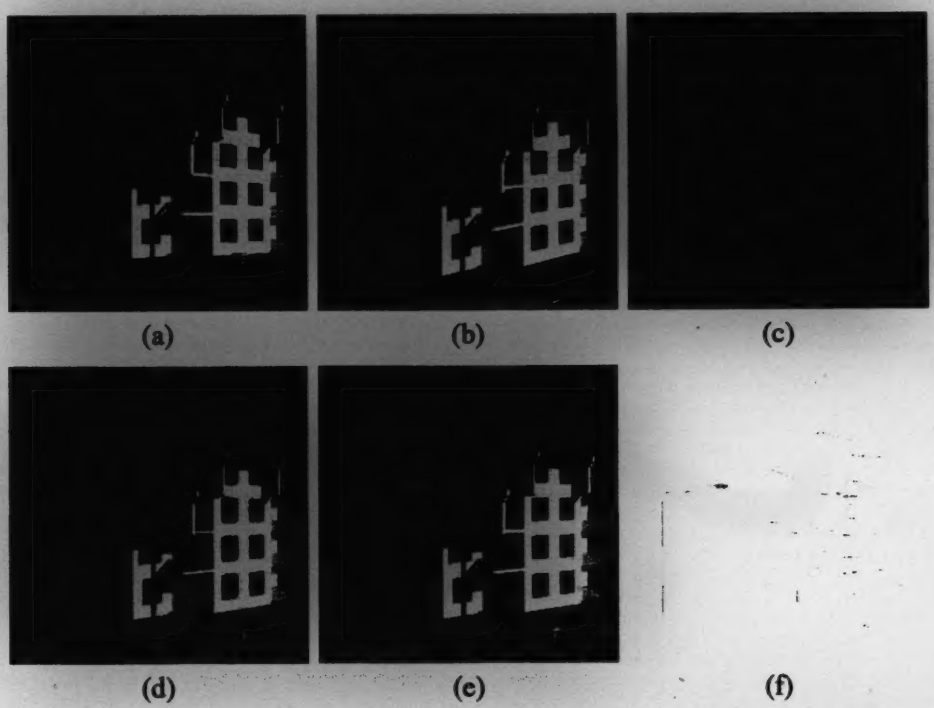


Figure 4-20: Novel view generated by morphing for an architecture scene.

About twenty regions are used to control different parts of the scene. These regions are separated into four layers, which are the street light and the garbage can, the building, the lawn, and the sky background. The hierarchical polynomial fit filter is used to interpolate the unknown part of each layer. Particularly, the hidden information of the sky background is recovered using both reference images. The recovered sky is shown in

Figure 4-20(c).

Figure 4-20(d) shows the morphing result generated for the in-between view. For comparison, the rendered image generated for the in-between view is shown in Figure 4-20(e). The absolute difference between rendered image and morphing result is shown in Figure 4-20(f). The darker the area the larger is the error.

4.3.5 Comparison with Feature-based Morphing

Like other morphing techniques, layer-based morphing can also be used to generate transformation between different objects. Here the results using the layer-based rendering approach are compared with that of the feature-based morphing approach.

Figure 4-21: Control features defined for feature-based morphing.

The two original images, one for a kitten and one for a puppy, are shown in Figure 4-21. It is noteworthy that the kitten's ears are pointing upward, while the puppy's ears are flapping downward and are mainly inside the puppy's outline. When using feature-based approach, the best that can be done is to define the control features along the outline of the kitten and of the puppy, as shown in Figure 4-21.

(a) (b) (c)

Figure 4-22: Feature-based morphing results.

The three in-between images generated by the feature-based morphing are shown in Figure 4-22(a), (b), and (c). The ghosting effect is noticeable at the right ear location in these in-between images.

Figure 4-23: Control features defined for layer-based morphing.

The layer-based approach gives more flexibility to generate the transition since different parts of the image can be morphed separately. Figure 4-23 shows the control features defined for the two source images. Green polygons indicate the regions defined, and red lines show additional features defined.

(d) (e) (f)

Figure 4-24: Layer-based morphing results.

Since separate regions are used to control the ears, the effect that the ears of the kitten are rotated down and change to the puppy's ears can be obtained without affecting nearby pixels. Three in-between frames generated are shown in Figure 4-24 (d), (e), and (f). It is also worth noting that the eyes and the nose in these frames are much sharper than those in the traditional transition since separate regions are used to control the morphing of the eyes and of the nose locally.

4.3.6 Discussions

Instead of interpolating between two images, layer-based morphing interpolates between two multi-layered images, which are generated from input reference images automatically. Hidden information within each layer is recovered using the hierarchical polynomial fit filter. Users do not need to edit the image directly to fill the holes, which is a cumbersome process. Since visibility problems are solved using multi-layered images, layer-based morphing can generate in-between novel views for much more complex scenes.

In addition, in layer-based morphing, different regions are used to control different objects in the scene. Control features have local effects only inside the associated regions. The warping function becomes many-to-one, and relative priorities are used to determine the final mapping. Consequently, layer-based morphing can morph different parts of the scene separately, and therefore, gives users more flexibility to control transitions.

For the same transition, features used by layer-based morphing are similar to that used by feature-based morphing. The only difference is that in feature-based morphing features are defined by line segments and in layer-based morphing some features are defined by polygons (boundaries of regions). Comparison between the two approaches shows that the results of layer-based morphing are better than that of feature-based morphing. However, extra human interactions are needed in layer-based morphing to specify the regions and to separate them into layers.

Chapter 5
Reliability-based Stereo Matching

To handle dynamic scenes, practical IBR approaches have to deal with sparsely sampled scenes, since it is not feasible to record and render sufficiently dense samples in real time. Previous research [22] has shown that depth information helps to reduce the sampling density requirement. However, capturing the actual depth information requires special equipment. Most equipment also has difficulty capturing depth information in real time, and therefore, cannot be used for dynamic scenes. To solve this problem, a novel stereo vision approach, called the reliability-based algorithm, is presented in this chapter, which can estimate accurate depth information based on captured images.

As reviewed in Section 2.3, the most accurate stereo vision techniques try to solve the stereo problem under the optimization framework. Different optimization techniques, such as dynamic programming [16] and graph cut [85], have been used to find the global optimized solution under some given parameters. However, due to the complex nature of the stereo vision problem, it is difficult, if not impossible, to have a universal set of parameters that can produce good disparity maps for different stereo images. In fact, even within a single image, it is highly possible that different regions should use different values of parameters since the signal-to-noise ratio may vary within the image. As a result, the best solution in terms of the given parameters may not necessarily be a good solution.

Here, the motivation is to incorporate a reliability measure into the optimization technique. The reliability measure is a novel method to evaluate the quality of the result. The new reliability-based algorithm is a relaxation process, which integrates a reliability

thresholding process and consistency constraints. The reliability thresholding process is based on dynamic programming, which is well known for its efficiency. It assigns disparities to pixels only when the reliability of the assignment is higher than a given threshold. The consistency constraints are defined to detect potential mismatches by cross-checking disparity maps generated for different views.

The organization of this chapter is as follows. In Section 5.1, two consistency constraints are introduced. The reliability thresholding process is discussed in Section 5.2. Section 5.3 explains how to integrate the reliability thresholding process with the consistency constraints. The experimental results are shown in Section 5.4.

5.1 Consistency of Disparity Maps

First of all, several concepts in stereo vision are re-formulated and generalized to facilitate the definitions of the constraints. The definitions of the two constraints and comparison with previously used constraints are given later.

5.1.1 The Stereo Vision Problem

Like many stereo vision algorithms, it is assumed that all given source images share the same image plane. The disparity of a pixel is defined using the inverse of the distance between the corresponding 3D point and the shared image plane [124]. As a result, pixels from different source images have the same disparity value if they are projections of the same 3D point in the scene. For two pixels with different disparity values, the one with a larger disparity value is the projection of a 3D point that is closer to the shared image plane.

Let $\mathbf{D}$ denote the value domain of disparity. Let $\mathbf{F}_s$ be the set that contains all the pixels in source image s (s=1…n), and let $\mathbf{G}_s$ be the set containing only pixels that have disparity values assigned. By definition, $\mathbf{G}_s \subseteq \mathbf{F}_s$ holds. A disparity map defined on image s is called a full solution, if $\mathbf{G}_s = \mathbf{F}_s$, and is called a partial solution otherwise.

A disparity map is defined as a function, $d^s: \mathbf{G}_s \rightarrow \mathbf{D}$, which assigns each pixel in $\mathbf{G}_s$ a disparity value. Assigning a disparity value d to pixel p in image s actually defines a point M_p^d in the 3D scene. It also defines a set $\mathbf{H}_p^d$, which contains the projections of M_p^d in all source images. Here, the set $\mathbf{H}_p^d$ is called a match and q ($q \neq p \wedge q \in \mathbf{H}_p^d$) the

corresponding pixel of p under disparity value d.

For any given image pair s and t, a warping function, $w^{s,t}:\mathbf{F}_s\times\mathbf{D}\rightarrow\mathbf{F}'_t$, is defined, which maps a pixel p in s to its corresponding pixel q in t under a given disparity value d. Here, $\mathbf{F}'_t\supseteq\mathbf{F}_t$ since q may be outside the boundary of t. Please note that no assumption is made on whether or not the epipolar lines coincide with image rows or columns. Instead, the warping functions, defined for different image pairs and different disparity values, satisfy the following two properties:

- One-to-one property: $w^{s,t}(w^{t,s}(p,d),d)=p$;
- Transitive property: $w^{s,t}(w^{r,s}(p,d),d)=w^{r,t}(p,d)$.

Consequently, the match $\mathbf{H}_p^d$ is a complete set of warping operations under disparity value d. This means that $\mathbf{H}_p^d=\mathbf{H}_q^d$ provided q is one of the corresponding pixel of p under disparity value d. It is also assumed that $M_p^d=M_q^d$, even though the 3D coordinates of the two may not be exactly the same. In the rest of this chapter, a match $\mathbf{H}$ and its corresponding 3D point M are used interchangeably.

5.1.2 Definitions of Constraints

Definition 1 (Strong Consistency Constraint): *A disparity map defined on $\mathbf{G}_s$ satisfies the strong consistency constraint if the following conditions hold:*

$$p\in\mathbf{G}_s\Rightarrow\forall t,\left(q\in\mathbf{G}_t\wedge d^s(p)=d^t(q)\right)\quad where\quad q=w^{s,t}\left(p,d^s(p)\right)$$

Equation 5-1: Strong consistency constraint.

Basically this constraint requires that if the disparity value of pixel p in image s is assigned, the disparity value of the corresponding pixel q in any image t must also be assigned. In addition, q must have the same disparity value as that of p. This indicates that the same 3D point M is visible from all these pixels.

Definition 2 (Weak Consistency Constraint): *A disparity map defined on $\mathbf{G}_s$ satisfies the weak consistency constraint if the following conditions hold:*

$$p\in\mathbf{G}_s\Rightarrow\forall t,\left(q\in\mathbf{F}_t\Rightarrow q\in\mathbf{G}_t\wedge d^s(p)\le d^t(q)\right)$$

Equation 5-2: Weak consistency constraint.

This constraint states that if the disparity value for pixel p is assigned, then the disparity value of the corresponding pixel q in any image t should also be assigned provided q is within the image boundary. Furthermore, q should either have the same disparity value as or a larger value than that of p.

Intuitively, the weak consistency constraint can be explained as follows: Assigning a disparity value to pixel p in image s defines a 3D point M_p^d. Since the corresponding pixel q is the projection of M_p^d in image t, M_p^d should be either visible at q or occluded by another 3D point, which is closer to the shared image plane. Either way, the disparity value at q should not be smaller than that of p. More formally, it can be proven that the weak consistency constraint holds for all scenes that can be represented by a set of 3D points.

Lemma 1: *The disparity maps for any given set of points in 3D space satisfy the weak consistency constraint, as long as the disparity maps share the same image plane and only one point is visible from any given pixel.*

Proof: Given any set of 3D points, one can generate the disparity maps for all images simultaneously by projecting the points in the order such that the one closest to the shared image plane is projected first. When projecting 3D point M to image s, one of the following three situations happens:

- $p \notin \mathbf{F}_s$, which means that the projection of M is not within the image boundary of image s. No change will be made to the disparity map of image s.
- $p \in \mathbf{F}_s \wedge p \notin \mathbf{G}_s$, which means that no previously handled point has been projected to pixel p, and therefore, M is visible at p. In this case, p is added to $\mathbf{G}_s$, and $d^s(p)$ is assigned according to the depth of M. Hence, assuming that the projection of M in any other image t is q, then $q=w^{s,t}(p,d^s(p))$. In addition, if M is also visible in image t, then $d^s(p)=d^t(q)$. Otherwise, $d^s(p) \leq d^t(q)$ since all previously projected points should not be farther away from the image plane than M.
- $p \in \mathbf{G}_s$, which means that at least one previously handled point has been projected to pixel p. Since only the closest point is visible from p, M will be occluded. No change will be made to the disparity map of image s.

As a result, after all points are projected, the generated disparity maps satisfy the

weak consistency constraint. ■

5.1.3 Comparison with Other Constraints

Both the uniqueness and ordering constraints are widely used in previous work [16, 194]. In this subsection, using a binocular matching scenario, the similarities and differences between these constraints and the weak consistency constraint are discussed. For binocular stereo, a match $\mathbf{H}_p^d$ contains two pixels, pixel p from the source image and q from the reference image. Hence, it can also be expressed as a pair $<p,q>$.

Each of the sub-figures in Figure 5-1 shows an un-skewed disparity space image for a pair of corresponding scanlines. Black denotes a match, gray denotes the match's inhibition zone, and striped denotes the match's occlusion zone.

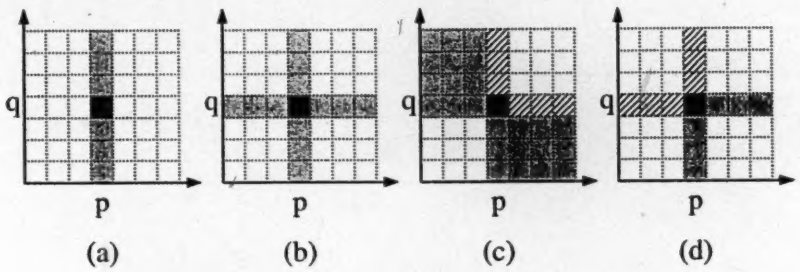


Figure 5-1: Effects of using different constraints.

The uniqueness constraint states that the disparity maps have a unique value per pixel [194]. As shown in Figure 5-1(a), if uniqueness is enforced in the source image only, a match $<p,q>$ will prohibit pixel p to be in any other matches. Consequently, all other matches in the same column are inhibited. If uniqueness is enforced in both the source and the reference images, as shown in Figure 5-1(b), a match will prohibit any other matches in the same row or the same column. This is the same as the strong consistency constraint defined earlier.

The ordering or monotonicity constraint states that if an object is to the left of another object in one stereo image, it is also to the left of the same object in the other image [16]. In practice, this constraint does not hold when thin foreground objects exist in the scene. As shown in Figure 5-1(c), if the ordering constraint is enforced, a match $<p,q>$ inhibits

matches $<u,v>$ if $u \geq p$ and $v < q$ or if $v \geq q$ and $u < p$.[1] Matches in the occlusion zone are allowed, but will be penalized by a predefined occlusion cost.

Finally, the effect of the weak consistency constraint is shown in Figure 5-1(d). The figure shows that under the weak consistency constraint, a match $<p,q>$ forbids matches $<p,v>$ ($v<q$) and matches $<u,q>$ ($u>p$) since these matches will occlude $<p,q>$. However, both p and q can be involved in another match that $<p,q>$ occludes, as long as the occlusion cost is applied. Since occlusion is modeled explicitly, the weak consistency constraint can be applied to the whole image, while the uniqueness and ordering constraints do not hold in either occluded areas or areas that contain thin foreground objects.

5.2 Reliability Thresholding Process

Here an efficient dynamic programming-based process is presented, which assigns disparity value d to pixel p only when the reliability of the corresponding match $\mathbf{H}_p^d$ exceeds a given threshold.

First of all, a reliability measure is defined for dynamic programming-based approaches in general. A match $\mathbf{H}_p^d$ is denoted as a pair $<p,d>$ for better readability in this section.

> ***Definition 3 (Reliability):*** *The reliability $R(p,d)$ of match $<p,d>$ is defined as the cost difference between the best path that does not pass through $<p,d>$ and the best path that passes through $<p,d>$.*

Obviously, if $<p,d>$ is on the best path that is found under no constraint, then $R(p,d) \geq 0$. The higher the value of $R(p,d)$, the more likely it is that the true disparity value of pixel p is d.

The reliability thresholding process is most closely related to the scanline optimization algorithm [148]. Comparing to the scanline optimization algorithm, the

---

[1] Please refer to figure 9 in Bobick and Intille's paper [16], which visualizes the inhibition zone in a skewed disparity space image.

reliability thresholding process has several improvements, which are elaborated below. It is noteworthy that since no attempt is made to determine occlusions within a scanline, both approaches do not require the scanlines to coincide with the epipolar lines.

5.2.1 Efficiency Improvement

The first improvement is at the implementation level. In Scharstein and Szeliski's implementation [148], the complexity of calculating each scanline is $O(L \times D^2)$, where L is the number of pixels per scanline, and D the disparity range. In the result reported, depending on the disparity range, the running time of the scanline optimization algorithm is 10%~60% slower than that of conventional dynamic programming algorithms that enforce ordering constraints.

In the present implementation, it is assumed that the same non-negative discontinuity cost, λ, is applied whenever neighboring disparity values are different, no matter how large the difference is. With this assumption, at most two possible paths need to be considered when searching for the best path that starts from match $<p,d>$. The first one connects to the path at $<p-1,d>$ so that no discontinuity cost will be incurred. The second path, if it differs from the first one, connects directly to the best path for the sub-problem that consists of the first $p-1$ pixels.

```
Initialize S[0,d], T[0,d], and m[0];
for ( p=1; p<Width; p++ ) {
  for ( d=0; d<_nRange; d++ ) {
    if ( m[p-1]==d || S[p-1,d]<S[p-1,m[p-1]]+λ )
      S[p,d]=S[p-1,d]+C[p,d], T[p,d]=d;
    else
      S[p,d]=S[p-1,m[p-1]]+λ+C[p,d], T[p,d]=m[p-1];
    if ( S[p,d]<S[p,m[p]] )
      m[p]=d;
  }
}
```

Figure 5-2: Algorithm for efficient cost calculation.

The new implementation proceeds as outlined in Figure 5-2. In the figure, C[p,d]

keeps the matching cost of <p,d>. When solving the sub-problem for the first p pixels, $\mathbf{S}[p,d]$ keeps the cost of the best path that starts from <p,d>, and $\mathbf{T}[p,d]$ keeps the corresponding transition. For every pixel p, $\mathbf{m}[p]$ keeps the value of d that has the smallest value of $\mathbf{S}[p,d]$.

As shown in Figure 5-2, since the search for the best path of the sub-problem is done only once, the complexity of the algorithm is cut down to $O(L \times D)$. The experimental results show that the speed of the new implementation is comparable with conventional dynamic programming algorithms.

5.2.2 Reliability Calculation

According to definition 3, calculating $R(p,d)$ for each match <p,d> on the best path requires running the dynamic programming algorithm again under the "do not pass" constraint. For efficiency concern, the approximate reliability $R'(p,d)$ is used instead and can be computed using the following algorithm.

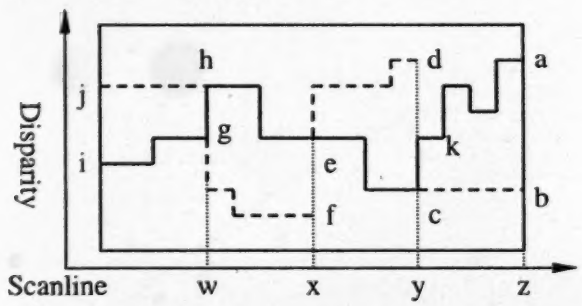


Figure 5-3: The best path and the alternate paths.

Besides arrays used in Figure 5-2, an additional array, $\mathbf{m}'[p]$, is used to keep the value of d that gives the second-to-the-smallest value of $\mathbf{S}[p,d]$. As shown in Figure 5-3, similar to traditional dynamic programming algorithms, the tracing starts from the rightmost pixel z. Assuming a=$\mathbf{m}[z]$ and b=$\mathbf{m}'[z]$, the best path (shown in solid lines) from "a" and an alternate path (shown in dashed lines) from "b" can be simultaneously traced. It is highly possible that the alternate path may merge with the best path in the trace. It can be proved that, using this algorithm, if the two paths merge at pixel p they will merge at <p,$\mathbf{m}[p]$>. This is because at least one of the two paths comes from <p+1,d>, where $d \neq \mathbf{m}[p]$. This path always connects to <p,$\mathbf{m}[p]$> since it is the only choice other than <p,d>.

As shown in the figure, assume that the two paths merge at pixel y and c=**m**[y], d=**m**′[y]. A new alternate path can start from d and continue the process. At the end of tracing, the best path and several alternate paths, one for each segment, are obtained. The approximate reliability R' of matches on the best path within each segment is then calculated using the cost difference between the corresponding alternate path and the best path at the end of the segment. For the above example, the approximate reliabilities of matches on *ka* is **S**[z,b]–**S**[z,a], and for those on *ec* is **S**[y,d]–**S**[y,c].

In Appendix A, the inequalities $R'-\lambda \le R \le R'+\lambda$ is proved. Therefore, when the discontinuity cost is small (normally within [0,4] in practice), using the approximate reliability measure instead of the reliability measure will not introduce too much bias.

It is noteworthy that when λ equals zero, the best path always passes through **m**[p] at pixel p, and the alternate paths always start from **m**′[p] and merge with the best path after one pixel. Both R(p,**m**[p]) and R'(p,**m**[p]) degenerate into measuring **S**[p,**m**′[p]]–**S**[p,**m**[p]], which is also equal to **C**[p,**m**′[p]]–**C**[p,**m**[p]] when λ equals zero.

```
best_id=second_id=m[_nWidth-1];
for ( p=_nWidth-1; p>=0; p-- ) {
  if ( best_id==second_id )
    second_id=m'[p], reliability=S[p,m'[p]]-S[p,m[p]];
  if ( reliability<_rThreshold )
    result[p]=-1;
  else
    result[p]=best_id;
  best_id=T[p,best_id], second_id=T[p,second_id];
}
```

Figure 5-4: Algorithm for tracing the best and alternate paths.

Figure 5-4 shows the pseudo-code for the tracing algorithm. By setting up the threshold, the algorithm selectively outputs some disparities on the best path that have enough conservative reliability. Since the best path and the alternate paths are traced simultaneously, the approximate reliabilities of all the matches on the best path can be calculated within the same pass. Therefore, very little computation overhead is added to the conventional algorithm.

5.2.3 Ground Control Points

Previous research has shown that pre-calculated ground control points help to eliminate mismatches by reducing the search space [16]. Here, they are also employed to improve efficiency. Whenever a pixel is selected as a ground control point in previous calculations, the reliability thresholding process will skip through it without any redundant computations.

```
if ( pixel p is a ground control point ) {
  Set d to be the disparity assigned for pixel p;
  if ( m[p-1]!=d && S[p-1,m[p-1]]+λ<S[p-1,d] )
    S[p,d]=S[p-1,m[p-1]]+λ+C[p,d], T[p,d]=m[p-1];
  else
    S[p,d]=S[p-1,d]+C[p,d], T[p,d]=d;
  m[p]=d;
}
```

Figure 5-5: Algorithm for handling ground control points.

As shown in Figure 5-5, the computational cost needed for handling ground control points is a constant. As a result, the complexity of the overall algorithm drops to $O(L_1+L_2\times D)$, where L_1 is the number of ground control points, L_2 the number of pixels to be calculated. This means that most of the calculations are spent on pixels with ambiguities. As more pixels are added into the ground control points, less computational time is needed.

5.3 Enforcing Consistency through Relaxation

This section explains how to integrate the reliability thresholding process with the consistency constraints through relaxation. Three major steps exist in each iteration: match suggestion, match validation, and disparity space update. All these steps can be done in polynomial time.

In the first step, the reliability thresholding process takes a reliability threshold and a discontinuity cost as inputs. It suggests some reliable matches for each of the source images. In the second step, the matches found through different images are validated

using either the strong or the weak consistency constraint, depending on the application. Each pixel involved in a confirmed match **H** will be added to solutions for the corresponding source image *s*, provided that **H** is visible in *s*. The pixels already added to the solution will be treated as ground control points by the reliability thresholding process in future iterations. Finally, in the last step, the disparity space is updated based on the weak consistency constraint using the new matches found in the current iteration. For each new match, the cost of matches in its inhibition zone is set to infinity. The cost of matches in its occlusion zone is set to the predefined occlusion cost, regardless of whether the original cost is higher or lower than the occlusion cost.

The disparity space update ensures that matches found in future iterations are weakly consistent with matches already confirmed, even though new matches may not be weakly consistent with each other. Since matches are always added to and never removed from the solutions, the algorithm is bounded to converge.

Depending on the application, different strategies can be used to combine the consistency constraints and the reliability thresholding process. Two different applications are discussed in the following subsections.

5.3.1 Unambiguous Stereo Matching

Due to difficulties of the binocular stereo matching problem, some researchers have investigated how to find unambiguous components of stereo matching [106, 144]. Following their ideas, how to utilize the reliability-based algorithm in this application is discussed next.

The ambiguities tend to appear in noisy areas, occluded areas, and weakly/periodic textured areas. Similar to previous approaches, occluded areas can be detected using consistency check, in particular, through the strong consistency constraint. Also, the reliability thresholding process can be used to identify weakly or periodic textured areas by setting up a reliability threshold. However, a unique feature of reliability thresholding process is that it can propagate the supports from reliable matches to their neighbors through the smoothness constraint. While previous techniques [144] can only increase the number of matches by lowering the threshold, which will result in more errors, it is now possible to increase the density of matches by increasing the discontinuity cost, without

lowering the reliability threshold. This helps to find correct matches in weakly or periodic textured areas, while still keeps mismatches caused by isolated noises out of the solution.

Following the idea above, the reliability-based algorithm proceeds as follows. Given a binocular stereo dataset, in the first stage, the reliability thresholding process is used to compute reliable matches under the strong consistency constraint, using a high reliability threshold, without any discontinuity cost. After the algorithm converges, which normally takes three to five iterations, the discontinuity cost is increased and another stage of iterations is started. Depending on how dense or how accurate the matches are needed, the user can choose the number of stages to use and the rate at which the discontinuity cost grows.

5.3.2 Multi-view Stereo Matching

The visibility problem cannot be fully addressed for binocular stereos. To generate full solutions, the best that can be done is to detect occluded areas and fill them using heuristic approaches. However, when multi-view stereo data is available, different algorithms [79, 85, 147] can be applied to better solve the problem.

As discussed in Subsection 2.3.4, the local approach [147] is fast but gives noisy results. The graph cut approach [85] produces very nice results but is slow. The reliability-based algorithm can fill the gap. Similar to the graph cut approach, in reliability-based algorithm, a separate disparity space $\mathbf{S}_{st}$ is initialized using the matching cost computed for each image pair $<s,t>$. In each iteration, based on $\mathbf{S}_{st}$, the reliability thresholding process calculates some reliable matches for both s and t. Assuming N pairs are used, the above calculation provides N suggested solutions for the center image s and one suggested solution for each of the peripheral images t. Even though there are redundant calculations for image s, they make it possible to detect mismatches that may pass both the reliability and the consistency tests.

In this application, the suggested matches for different images are validated by enforcing the strong consistency constraint between each pair and the weak consistency constraint among all images. That is, to confirm a match $\mathbf{H}$, which is found for pixel p in image s through image pair $<s,t>$, $\mathbf{H}$ must be visible in both s and t. In addition, $\mathbf{H}$ cannot occlude any existing or suggested matches in any other image u, but it is allowed that $\mathbf{H}$

is occluded in u.

After the match **H** is confirmed, it will be projected to all images. Assuming the projection of **H** is pixel q in image u. q will be added to $\mathbf{G}_u$ if it is not already in there. In addition, the disparity space $\mathbf{S}_{uv}$ for different image pair $\langle u,v \rangle$ are also updated using the weak consistency constraint.

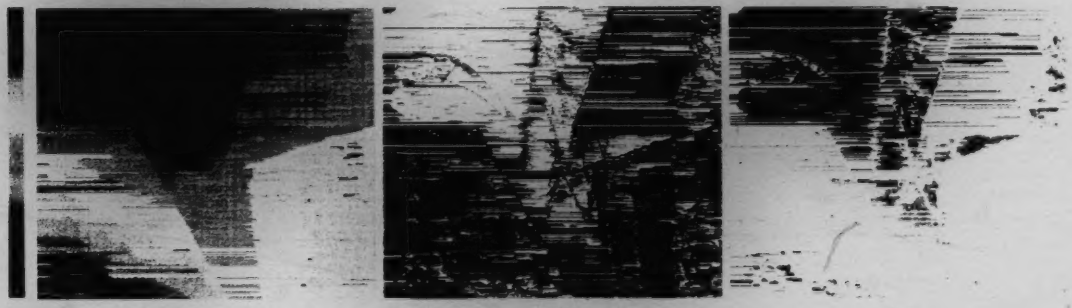
After the iteration converges, a disparity map for each source image is obtained. Disparity values are assigned to most of the pixels that are visible in both the center image and at least one of the peripheral images. As a result, the disparity map for the center image is normally very dense since very few pixels are not visible in any of the peripheral images. Those for the peripheral images are also denser than the matching results generated using only one pair since reliable matches found from other pairs are used. In reliability-based algorithm, missing disparity values are filled using median filtering under the weak consistency constraint.

5.4 Experimental Results

The experiments are conducted to evaluate the reliability-based algorithm presented in this chapter. The reliability thresholding process is tested first. The experimental results for the unambiguous stereo matching problem and the multi-view stereo matching problem are given later.

5.4.1 Reliability Thresholding

The "Venus" dataset [148] is used here. To make the results comparable with that reported for the scanline optimization algorithm, the same set of parameters is used here.



(a) (b) (c)

Figure 5-6: Results of the reliability calculation and thresholding.

Figure 5-6(a) shows the disparity map generated using the reliability thresholding process without setting the reliability threshold. The color coding used is shown beside the figure and is the same as that used by Sara [144]. Since no ground control points are utilized, the result is similar to that of the scanline optimization algorithm. However, the efficiency is improved. On an Athlon 1.5GHz PC with 1GB memory running Windows XP Professional, the current implementation can generate the above disparity map in 0.06 seconds. Scharstein and Szeliski's algorithm is re-implemented and it takes 0.56 seconds on the same system. Please note that the time needed for the matching cost computation is not included because it is the same for both algorithms. In addition, the matching costs need to be calculated only once when the reliability thresholding process is used in relaxation.

Figure 5-6(b) visualizes the approximate reliabilities calculated for different regions of the disparity map. The brighter the color, the higher the approximate reliability of the corresponding disparity value is. The result shows that the approximate reliabilities are low in textureless areas, such as the top-right corner. Referring to Figure 5-6(a), it is noteworthy that there are many horizontal streaks in this area. On the other hand, in areas where correct and smooth disparity values are produced, such as the top-left corner, the approximate reliabilities are relatively high.

After the reliability threshold is setup, the reliability thresholding process can give a partial solution that contains only reliable matches. As shown in Figure 5-6(c), most of the errors and streaks are removed in the partial solution. However, there are still some errors left in the result. This is partly because the large smoothness weight used (50 as suggested by Scharstein and Szeliski [148]) increases the difference between the reliability and the approximate reliability. Normally a much smaller smoothness weight is used since the reliability-based algorithm does not depend on the smoothness constraint to remove mismatches.

5.4.2 Unambiguous Stereo Matching

The second experiment compares unambiguous matches generated using Sara's approach [144] and the reliability-based algorithm. The four standard binocular datasets [148] are used for evaluation.

In the reliability-based algorithm, the same set of parameters is used for all datasets: The process discussed in Subsection 5.3.1 is run for three stages, the value of the reliability threshold is fixed at 2 throughout the process, and the value of λ steps through 0, 1, and 2 for the three stages. The Sara's approach is evaluated based on the published results [143].

Table 5-1: Comparison for unambiguous stereo matching

	Head and lamp		Sawtooth		Venus		Map	
	D (%)	*e* (%)	*D* (%)	*e* (%)	*D* (%)	*e* (%)	*D* (%)	*e* (%)
Stage 1	21.7	0.24	26.8	0.11	14.6	0.02	29.2	0.02
Stage 2	36.5	0.33	47.7	0.19	27.5	0.12	43.5	0.03
Stage 3	85.7	1.07	85.0	0.41	67.1	0.51	60.8	0.09
Sara's	45.7	2.05	61.7	2.15	47.6	1.54	69.9	0.76

The density (*D*) and error rate (*e*) of matches produced are shown in Table 5-1. The density is defined as the percentage of matches generated, and the error rate the percentage of bad matches (not within correct disparity ±1). Please note that, different from the measure used by Scharstein and Szeliski [148], bad matches within occluded areas are also counted in the error rate calculation. This generates slightly higher error rates for both approaches. It is believed that this is a more appropriate evaluation for unambiguous stereo matching applications, which are supposed to detect occluded areas.

The results show that with different datasets, the reliability-based algorithm gives a lower error rate. In addition, after three stages, in the first three datasets, it can produce denser disparity maps as well. The result for the "Map" dataset is not as dense as the other datasets because, as already shown [148], this dataset requires a higher smoothness weight than the other three. If the value of λ is increased to 4 and the algorithm is run for another stage, disparity maps with 76.9% density and 0.17% error rate can be obtained.

The disparity map generated and the corresponding bad pixels for two of the datasets are shown in Figure 5-7. The first, second, and third column in the figure show the results after each of the three stages, respectively. The results clearly illustrates how weakly textured areas are filled up as the discontinuity cost is increased.

Figure 5-7: Unambiguous stereo matching results.

Using the reliability-based algorithm to generate reliable matches is also efficient. The experiments show that the three-stage process normally requires a total of 10~15 iterations. In practice, the CPU time needed is between 2~5 seconds for each of the above datasets. If necessary, the computation time can be reduced at the cost of lowering the density or the accuracy of matches by limiting the number of iterations per stage.

5.4.3 Multi-view Stereo Matching

Finally, the reliability-based algorithm is tested for the multi-view stereo matching problem. Two datasets used by Satoh and Ohta [147] is used in the experiment. Each dataset contains 9 images.

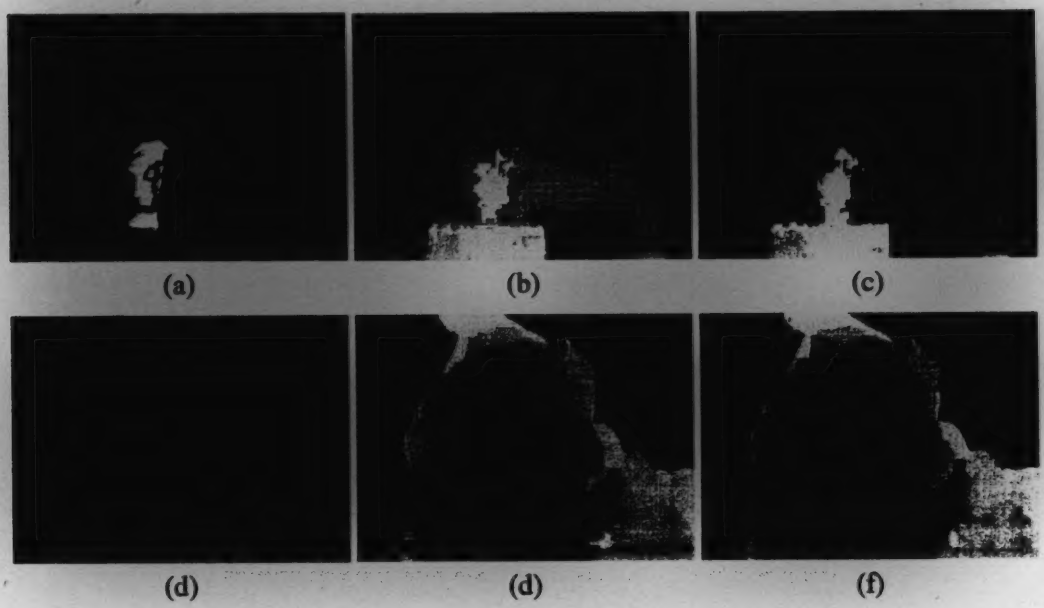


Figure 5-8: Trinocular and multi-view stereo matching results.

Figure 5-8(a) and (d) shows the center images of the two datasets used. To evaluate the performance of the reliability-based algorithm on trinocular datasets, three of the nine source images (center, top, and right) are used as input first. The generated disparity maps are shown in Figure 5-8(b) and (d). The algorithm is then evaluated using five images (center, top, bottom, left, right), and the results are shown in Figure 5-8(c) and (f).

Table 5-2: Comparison for multi-view stereo matching

	# of images	# of pairs	e (%)	E (%)	Time (sec)
Reliability-based	3	2	2.03	13.31	2+6
	5	4	1.86	12.62	4+11
Stereo by Eye Array	9	8	4.83	23.45	9+0.01
Graph cut	5	4	2.75	6.13	369
	5	10	2.30	4.53	837

The comparison using the first dataset, which has the ground truth available, is shown in Table 5-2. In the table, the mismatch rate (E) is defined as the percentage of pixels that do not match exactly with the ground truth. The running time is reported in the form: matching cost computation time + disparity computation time.

The results of the Stereo by Eye Array approach are based on a local implementation using 3×3 matching windows and "new mask" as the detection algorithm. The results of graph cut are reported by Kolmogorov and Zabih [85]. In their work, the time is measured on a 450MHz UltraSPARC II. It is noteworthy that the same five images are used in the graph cut approach as in the reliability-based algorithm, while nine images are used in the Stereo by Eye Array approach.

Table 5-2 shows that the results using the reliability-based algorithm have less than half of the bad pixels as that using the Stereo by Eye Array approach, even though less image pairs are used. In addition, within the reported time, the new approach generates the disparity maps for all source images, while the Stereo by Eye Array approach only produces the center one. Compared with the results of the graph cut approach, the results using the reliability-based algorithm have a slightly lower error rate and a much higher mismatch rate. However, it seems the computation cost of the new approach is much lower, even though no comparison on the same platform is available.

Chapter 6
Rayset Application for Dynamic Scenes

Dynamic IBR is a new research topic on how to generate new videos based on videos captured at fixed viewpoints, so that one can observe dynamic events from arbitrary viewpoints. Compared with static IBR, it has the following requirements:

- The capturing process has to be done in real time although the rendering process can be done offline.
- The capturing devices cannot disturb people in the scene and cannot change the lighting condition in the scene either.
- The rendering process should be automatic. No human interactions should be required.
- The rendering algorithm has to deal with a sparse set of viewpoints, which are captured by a limited number of cameras.

It is noteworthy that, as a result of the above requirements, most depth capturing devices cannot be used in a dynamic scene. For example, a range scanner cannot acquire the depth information in real time. The structured light method will change the lighting condition in the scene. As a result, it is difficult, if not impossible, to capture accurate depth information for dynamic scenes.

Also due to the above requirements, most IBR approaches can hardly be extended to dynamic scenes. For example, plenoptic sampling approaches [56, 70, 97, 166] require dense samples to produce unlimited novel views. The majority of depth warping approaches [115, 125, 133] need accurate depth information. Almost all texture mapping approaches [42, 126, 186] involve human interactions or measuring devices to provide

3D geometric proxies, and all scene morphing approaches [161] involve human interactions to provide 2D control information.

In this chapter, a novel rayset-based approach, called camera field rendering, is discussed. This approach satisfies the requirements listed above and can therefore be applied to dynamic scenes. In the rest of this chapter, the parameterization scheme is presented in Section 6.1. Two interpolation techniques are introduced in Section 6.2 and 6.3, respectively. Section 6.4 presents the experimental results for static scenes and Section 6.5 presents those for dynamic scenes.

6.1 Camera Field Parameterization

The parameterization scheme used in camera field rendering is discussed in this section. The general parameterization scheme is illustrated first, followed by two special cases.

6.1.1 General Parameterization Scheme

In general, a camera field is defined on a surface, called the support surface. Under the rayset taxonomy, the parameterization scheme can be defined by the following support function:

$$\begin{cases} \mathbf{S}_x(u,v,s,t) = \mathbf{F}_x(u,v) \\ \mathbf{S}_y(u,v,s,t) = \mathbf{F}_y(u,v) \\ \mathbf{S}_z(u,v,s,t) = \mathbf{F}_z(u,v) \\ \mathbf{S}_\theta(u,v,s,t) = \Theta\left(f \cdot \mathbf{n}(u,v) + s\left.\frac{\partial \hat{\mathbf{F}}(p,q)}{\partial p}\right|_{u,v} + t\left.\frac{\partial \hat{\mathbf{F}}(p,q)}{\partial q}\right|_{u,v} \right) \\ \mathbf{S}_\phi(u,v,s,t) = \Phi\left(f \cdot \mathbf{n}(u,v) + s\left.\frac{\partial \hat{\mathbf{F}}(p,q)}{\partial p}\right|_{u,v} + t\left.\frac{\partial \hat{\mathbf{F}}(p,q)}{\partial q}\right|_{u,v} \right) \end{cases}$$

Equation 6-1: Support function for camera fields.

where $\mathbf{F}(u,v)$ is the parametric equation of the support surface, f the focal length of the camera, and $\mathbf{n}(x,y)$ the unit normal of the support surface at (x,y). The partial derivatives are normalized. Subscripts denote the corresponding components.

Obviously, to sample such a rayset, a 2D array of pinhole cameras can be arranged on the support surface, and adjusted so that their viewing directions are normal to the

support surface. When the cameras are aligned accurately enough, images captured under such a setting form a camera field directly. Otherwise, a rectification process may be required.

6.1.2 Special Cases

Depending on the scene or object to be sampled, different kinds of support surfaces can be selected, e.g. planes, parametric surfaces, or free-form surfaces. In particular, if a plane is used, it becomes a planar camera field. The support function shown in Equation 6-1 degenerates into:

$$\begin{cases} \mathbf{S}_x(u,v,s,t) = \mathbf{F}_x(u,v) \\ \mathbf{S}_y(u,v,s,t) = \mathbf{F}_y(u,v) \\ \mathbf{S}_z(u,v,s,t) = \mathbf{F}_z(u,v) \\ \mathbf{S}_\theta(u,v,s,t) = \Theta(f \cdot \mathbf{n} + \mathbf{F}(s,t) - \mathbf{F}(0,0)) \\ \mathbf{S}_\varphi(u,v,s,t) = \Phi(f \cdot \mathbf{n} + \mathbf{F}(s,t) - \mathbf{F}(0,0)) \end{cases}$$

Equation 6-2: Support function for planar camera fields.

where $\mathbf{F}(u,v)$ is the parametric equation of the plane and $\mathbf{n}$ the normal of the plane.

It is noteworthy that since the camera matrix uses a 2D array of cameras on a plane to capture real scenes [146], the datasets obtained naturally form the planar camera field.

Different parametric surfaces, such as a cylinder, a sphere, and a cone, can also be used. A cylinder is used here as an illustration. For a cylindrical camera field, the support function can be derived from Equation 6-1 or from the geometry directly:

$$\begin{cases} \mathbf{S}_x(u,v,s,t) = \mathbf{C}_x + R \cdot \cos(u) \\ \mathbf{S}_y(u,v,s,t) = \mathbf{C}_y + R \cdot \sin(u) \\ \mathbf{S}_z(u,v,s,t) = \mathbf{C}_z + v \\ \mathbf{S}_\theta(u,v,s,t) = u + \arctan(s/f) + k\pi \\ \mathbf{S}_\varphi(u,v,s,t) = \arctan\left(t/\sqrt{f^2+s^2}\right) \end{cases}$$

Equation 6-3: Support function for cylindrical camera fields.

where $\mathbf{C}$ is the center of the cylindrical camera field and R the radius of the cylinder. When k=0, the cameras face outward and sample the environment. When k=1, the cameras face the inside of the cylinder and sample the object as well as the environment is behind the object.

Please note that the cylindrical camera field is different from the cylindrical panorama. The latter defines the image plane on a cylinder and is a 2D rayset, while the former defines the centers of the projections on a cylinder and is a 4D rayset.

6.2 Color-matching Interpolation

First, let us try to interpolate between sparse views without explicit knowledge of depth information. The corresponding approach, which searches for a physical point in the scene along a testing ray, is discussed in the rest of this section.

6.2.1 The Interpolation Problem

Here the interpolation problem is illustrated using a planar camera field. The problem for cylindrical camera field is similar.

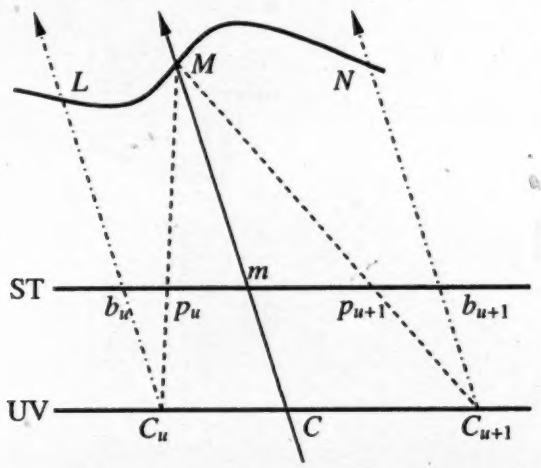


Figure 6-1: Interpolation between reference images for planar camera field.

Figure 6-1 shows the cross-section of a planar camera field. Assume that two cameras are set up at locations C_u and C_{u+1} with their viewing directions perpendicular to the support plane (UV). A novel ray, Cm, intersects the support plane at C and the image plane (ST) at m. Using linear interpolation, the illumination at m is given by:

$$I = \frac{C_u C}{C_u C_{u+1}} \times I_u(b_u) + \frac{C C_{u+1}}{C_u C_{u+1}} \times I_{u+1}(b_{u+1})$$

Equation 6-4: Interpolation on camera plane.

where $I_u(x)$ denotes the intensity of pixel x in image u.

Obviously, if the support plane is not sampled densely enough, the interpolation result will be blurry because it is obtained by interpolating between physically different points, in particular, point L and N, in the scene. As shown in Figure 6-1, the intersection between ray Cm and the object is point M, whose projections on image u and image $u+1$ are p_u and p_{u+1}, respectively. Hence, a better result is obtained if these two pixels are used for interpolation, i.e.:

$$I = \frac{C_u C}{C_u C_{u+1}} \times I_u(p_u) + \frac{C C_{u+1}}{C_u C_{u+1}} \times I_{u+1}(p_{u+1})$$

Equation 6-5: Interpolation between projections of the same physical point.

Now the question is how to determine the locations of pixel p_u and p_{u+1}, which are the projections of the same point, M, in the scene. In the color-matching interpolation, this is done by searching physical points in the scene. First, the following two assumptions are required:

- Any point in the scene that is visible from the novel viewpoint is also visible in four nearby cameras.
- The projections of the same physical point in the scene should have a higher level of color consistency than the projections from different physical points.

If the above assumptions hold, then along the testing ray, the projection of the intersection should have the highest level of consistency in color, i.e. have the smallest dissimilarity[1]. Hence, the point on the testing ray, whose projections to nearby reference images give the smallest dissimilarity, is the required intersection.

The above assumptions do not always hold in the real world. The first assumption is invalid around occluding boundaries in the scene. The second assumption fails for regions without textures. However, the experimental results suggest that the presented approach is not too sensitive to the violation of the two assumptions.

6.2.2 Matching along Epipolar Lines

Projecting 3D points on the novel ray to reference images involves a lot of computations.

---

[1] The dissimilarity between two pixels is defined as the Euclidean distance between the intensities of the pixels in the RGB color space.

However, based on the properties of the camera field parameterization scheme and the epipolar constraint, an efficient search algorithm, which works in the 2D image space, can be used. The general searching algorithm is the same even though the equations used depend on the format of the support surface. In the following, the planar and cylindrical camera fields are discussed first, and then the pseudo-code for the general searching algorithm is presented.

As shown in Figure 6-1, the vanishing point for ray Cm on reference image C_u is b_u. Without loss of generality, it is assumed that the intersection is in front of the focal plane. This is definitely true with a real camera since all the real objects captured in the images are in front of the camera. For synthetic scenes, however, the image plane must be placed carefully to make sure that no virtual object is between the image plane and the center of projection.

Under the above assumption, the search is limited to within the interval $b_u m$ for reference image u. In addition, the following can be derived based on similar triangles shown in Figure 6-1:

$$\frac{Cm}{CM} = 1 - \frac{mM}{CM} = 1 - \frac{p_u m}{C_u C} = \frac{b_u p_u}{C_u C}$$
$$\frac{Cm}{CM} = 1 - \frac{mM}{CM} = 1 - \frac{p_{u+1} m}{C_{u+1} C} = \frac{b_{u+1} p_{u+1}}{C C_{u+1}}$$

Equation 6-6: Constraint between the projections for planar case.

Now a distance function $E_u(p_u)$ is defined as:

$$E_u(p_u) = \frac{1}{CM} = \frac{b_u p_u}{C_u C \times Cm}$$

Equation 6-7: Distance function for planar case.

Therefore, if different pixels p_u from different reference images u are the projections of the same point on the testing ray Cm, the function $E_u(p_u)$ must have the same value for different u.

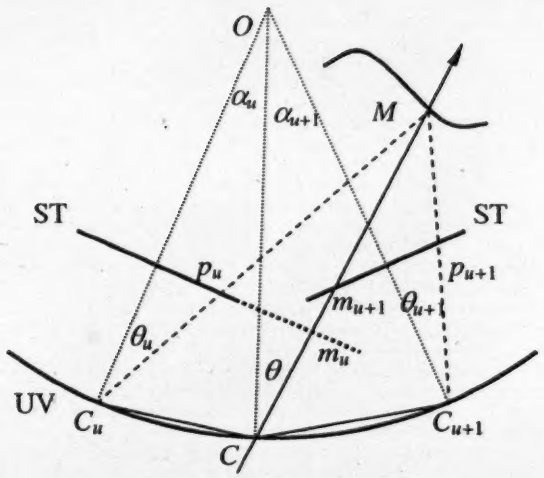


Figure 6-2: Interpolation between reference images for cylindrical camera field.

Similar relations can also be deduced for the cylindrical camera field. Figure 6-2 shows the projection of a cylindrical camera field on the X-Y plane. Using triangles C_uCM and $CC_{u+1}M$ the following relations can be derived:

$$\frac{CM}{\sin((\pi-\alpha_u)/2-\theta_u)}=\frac{C_uC}{\sin(\alpha_u-\theta+\theta_u)}$$
$$\frac{CM}{\sin((\pi-\alpha_{u+1})/2+\theta_{u+1})}=\frac{CC_{u+1}}{\sin(\alpha_{u+1}+\theta-\theta_{u+1})}$$

Equation 6-8: Constraint between projections for cylindrical case.

Therefore, $E_u(p_u)$ is defined as:

$$E_u(p_u)=\frac{1}{CM}=(-1)^k\frac{\sin\left((-1)^k\alpha_u+\theta_u-\theta\right)}{C_uC\times\cos\left((-1)^k\alpha_u/2+\theta_u\right)}$$

Equation 6-9: Distance function for cylindrical case.

where k is equal to 0 when C is on the right hand side of C_u, and k is equal to 1 otherwise.

As a result, what needs to be done is to simultaneously move the current pixel, p_u, along the epipolar line and keep $E_u(p_u)$ the same for all nearby reference images u. The pixels that have the smallest dissimilarity are the projections of the desired intersection. In practice, for 2D images, the current pixels for nearby reference images are moved along the discrete approximations of the epipolar lines one pixel at a time, while different $E_u(p_u)$ are kept roughly the same. This is done by finding the pixel that has the highest value of $E_u(p_u)$ and by moving it closer to b_u. The pseudo-code for this algorithm is

shown in Figure 6-3.

```
For each nearby reference image u
  Set p[u] = m_u, E[u] = E_u(p_u), Color[u] = I_u(p_u);
While ( true ) {
  Set mean = weighted-average of Color[u] for different
u;
  Set error = weighted-sum of the Euclidean distances
between Color[u] and mean;
  If ( error < min_error )
    Update best_match = mean, min_error = error;
  Find the image u that has the highest value of E[u];
  Move p[u] one pixel closer to the b_u;
  If ( b_u p_u<0 ) Break;
  Update E[u] = E_u(p_u), Color[u] = I_u(p_u);
}
return best_match;
```

Figure 6-3: Algorithm for color-matching interpolation.

6.2.3 Result of Color-matching

To evaluate the interpolation results, four reference images from the "head and lamp" dataset are used here to interpolate the novel view, whose viewpoint is at the center of the square that is formed by the viewpoints of the four reference images.

(a) (b)

Figure 6-4: Interpolation results (a) linear (b) color-matching.

Figure 6-4(a) shows the result generated using linear interpolation. Since all reference

images have equal weights, the generated result is the same as cross-blending the four reference images together. Figure 6-4(b) is generated using color-matching interpolation and is much sharper.

As shown in Figure 6-4(b), artifacts still exist, mainly because the two assumptions are not satisfied. Many details are lost in the rendered image, such as text on the poster and highlights on the lamp. This is because of the violation of the second assumption, i.e., the projections of a physical point do not have the smallest dissimilar value.

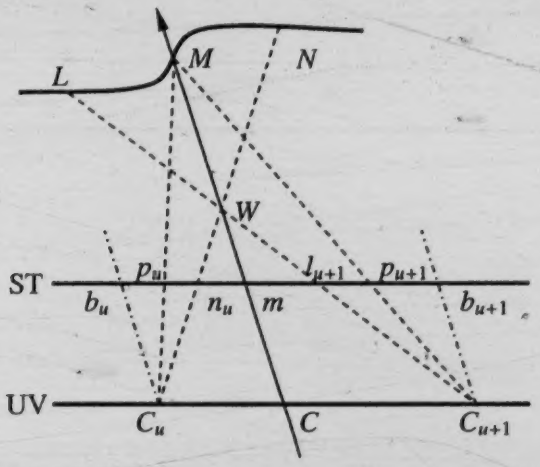


Figure 6-5: Incorrect interpolations using color-matching interpolation.

Figure 6-5 uses a planar camera field to illustrate the cause of this problem. As shown in the figure, ray Cm intersects the surface at location M, and the projections of M on the two images are p_u and p_{u+1} respectively. Ray $C_u n_u$ intersects the surface at location N, ray $C_{u+1} l_{u+1}$ intersects the surface at location L, and their intersection, W, lie on ray Cm. In cases that the dissimilarity between n_u and l_{u+1} is smaller than that of p_u and p_{u+1}, the second assumption supposes that there exists a point in the scene at location W, which generates the observed projections n_u and l_{u+1}. Actually, in the above example, even though the two dissimilarities are the same, the false target can still be selected since, as compared with M, location W is closer to the viewpoint.

6.3 Disparity-searching Interpolation

Now let us try to solve the above limitation using the depth information estimated in Chapter 5. Similar to that is discussed in Chapter 5, the disparity is defined based on the

inverse distance between the corresponding 3D point and the image plane. In particular, it is defined as the following function:

$$\delta(m) = \frac{Cm}{CM}$$

Equation 6-10: Definition of disparity.

where C is the center of projection, m the pixel on the image plane, and M the intersection between ray Cm and the object in the scene.

Since all the intersections are assumed to be in front of the image plane, the legal value of disparity with this definition is between zero and one.

6.3.1 Searching Based on Disparities

Suppose relatively accurate disparity information can be obtained. Now, the question is how to interpolate multiple images with disparities. Here a backward searching approach is presented. In the rest of this subsection, the cases for planar and cylindrical camera fields are discussed first, followed by the general searching algorithm.

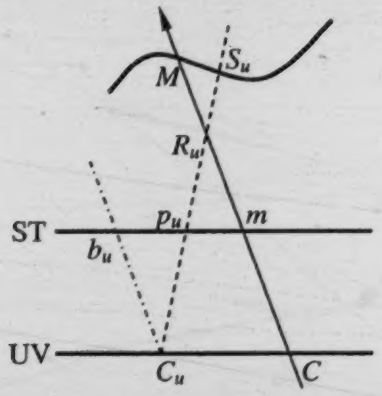


Figure 6-6: Intersection searching using disparity for planar camera field.

Figure 6-6 shows the cross-section of a planar camera field. Suppose it is required to determine the intersection of ray Cm with objects in the scene using a known view, whose center of projection is C_u. As was mentioned above, the image of intersection M lies on segment b_um. For any pixel p_u on b_um, the length of C_uR_u is calculated using the following equation:

$$\frac{C_u p_u}{C_u R_u} = \frac{b_u p_u}{C_u C} \Rightarrow C_u R_u = C_u p_u \times \frac{C_u C}{b_u p_u}$$

Equation 6-11: Distance between viewpoint and testing ray for planar case.

Also, based on the known disparity map, the length of $C_u S_u$ is given by:

$$\delta_u(p_u) = \frac{C_u p_u}{C_u S_u} \Rightarrow C_u S_u = C_u p_u \times \frac{1}{\delta_u(p_u)}$$

Equation 6-12: Distance between viewpoint and surface for planar case.

Now an intersection searching function $F_u(p_u)$ is defined as:

$$F_u(p_u) = C_u p_u \times \left(\frac{1}{C_u S_u} - \frac{1}{C_u R_u} \right) = \delta_u(p_u) - \frac{b_u p_u}{C_u C}$$

Equation 6-13: Intersection searching function for planar case.

Obviously, if the test ray intersects the surface at location M and its projection on image u is p_u, then function $F_u(p_u)$ should be zero. Otherwise, $F_u(p_u)>0$ means that the intersection with the surface is in front of the intersection with the testing ray, and $F_u(p_u)<0$ means that the intersection with the surface is behind the intersection with the testing ray. Therefore, the problem of searching the corresponding point is equivalent to the problem of finding the zero-crossing point of $F_u(p_u)$. $F_u(p_u)$ can be efficiently evaluated since it only needs one addition and one division operation.

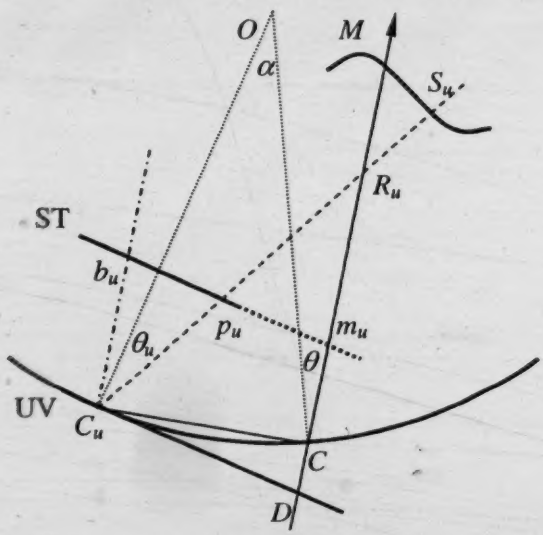


Figure 6-7: Intersection searching using disparity for cylindrical camera field.

A similar function can be found for the cylindrical camera field as well. In Figure 6-7, the length of C_uR_u can be calculated by:

$$\frac{C_u p_u}{C_u R_u} = \frac{b_u p_u}{C_u D} \Rightarrow C_u R_u = C_u p_u \times \frac{C_u D}{b_u p_u}$$

Equation 6-14: Distance between viewpoint and testing ray for cylindrical case.

Since

$$\frac{C_u C}{\sin((\pi+\alpha)/2-\theta)} = \frac{C_u D}{\sin(\pi/2-\alpha+\theta)} \Rightarrow C_u D = \frac{C_u C \times \cos(\alpha/2-\theta)}{\cos(\alpha-\theta)}$$

Equation 6-15: Distance between viewpoint and surface for cylindrical case.

$F_u(p_u)$ becomes:

$$F_u(p_u) = C_u p_u \times \left(\frac{1}{C_u S_u} - \frac{1}{C_u R_u}\right) = \delta_u(p_u) - \frac{b_u p_u}{C_u C} \times \frac{\cos(\alpha-\theta)}{\cos(\alpha/2-\theta)}$$

Equation 6-16: Intersection searching function for cylindrical case.

$F_u(p_u)$ is not continuous since the disparity function $\delta_u(p_u)$ is defined on discrete samples. In order to find the zero-crossing point, linear interpolation can be used to reconstruct the continuous function. Since linear interpolation will also connect pixels that belong to different objects together, it will generate "rubber sheets" in the resulting images, in which different objects are stretched inappropriately. The following scenario illustrates the cause of rubber sheets.

As shown in Figure 6-8, ray *Cm* intersects the background surface at location *M*, which is projected at location p_{u-1} in image C_{u-1} but not visible in image C_u. Two pixels n_u and l_u are adjacent to each other in image C_u. Ray $C_u n_u$ intersects the background surface at location *N*, and ray $C_u l_u$ intersects the foreground surface at location *L*. Under such a scenario, $F_u(l_u)>0$ and $F_u(n_u)<0$. Therefore, linear interpolation will give a zero-crossing point, which indicates that the intersection is at location *W*, where *W* is between *N* and *L*. Since location *W* is closer than location *M*, the color for ray *Cm* will be computed by interpolating point *L* and *N*, which is not correct.

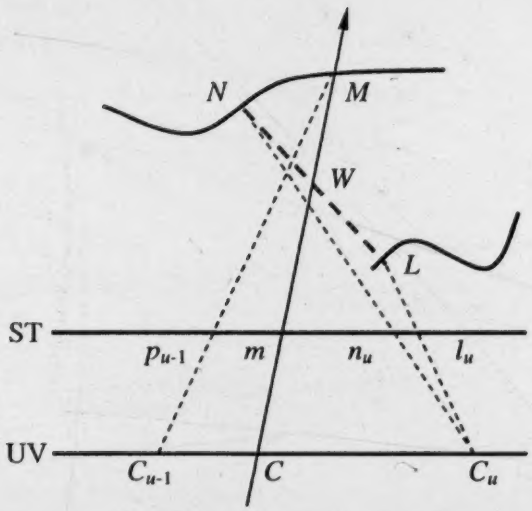


Figure 6-8: "Rubber sheets" problem caused by linear interpolation.

To remove the rubber sheets effects, a threshold t is defined. Whenever $|F_u(x)-F_u(x+1)|>t$, it is assumed that pixels x and $x+1$ are the projections of points on two different objects. Therefore, no intersection will be computed any more.

The above discussion is for a single reference image. When multiple reference images are available, the closest zero-crossing point among different function, $F_u(x)$, for different reference image u is determined. The corresponding pixel will be used to color the novel ray. In the case when two or more zero-crossing points have the same closeness to the center of projection, the final color is produced through weighted-averaging the corresponding pixels according to the distances between the reference views and the novel view.

It is noteworthy that there is no need to search for all the zero-crossing points and project them back to three dimensions to find which one is closest. It is known that a point on Cm that is closer to the center of projection is projected to a pixel that is closer to m. Therefore, the current pixel, p_u, can be moved along the epipolar line while $E_u(p_u)$ is kept the same for all nearby reference image u. The first zero-crossing point will be the closest intersection. The pseudo-code of the algorithm is shown in Figure 6-9.

```
For each nearby reference image u
  Set p[u] = m_u, E[u] = E_u(p_u), NewF[u] = F_u(p_u);
While ( true ) {
  Find the image u that has the highest value of E[u];
  Set OldF[u] = NewF[u], NewF[u] = F_u(p_u);
  If ( NewF[u]×OldF[u]<0 && abs(NewF[u]-OldF[u])>t )
    Return color interpolation between I_u(p_u) and
I_u(oldp_u);
  Set Oldp[u] = p[u];
  Move p[u] one pixel closer to the b_u;
  If ( b_u p_u<0 )
    Return "no intersection found";
}
```

Figure 6-9: Algorithm for disparity-searching interpolation.

6.3.2 Result of Disparity-searching

The same "head and lamp" dataset is used for evaluation. The disparity information is computed using the reliability-based algorithm described in Chapter 5.

(a) (b)

Figure 6-10: Interpolation results (a) disparity-searching (b) combined.

Figure 6-10(a) shows the rendering result of the disparity-searching interpolation. The result demonstrates that the details of the scene, e.g. highlights on the lamp and texts on the poster, are preserved. However, since a tight threshold is used to remove the "rubber sheet" effect, the disparity-searching interpolation fails to find the correct intersections in some areas, which are marked by green pixels in Figure 6-10(a).

The above problem is caused by errors in the disparity maps. The strategy of providing a robust rendering algorithm is to combine the two interpolation techniques together. That is, try to search for an intersection using the disparity-searching interpolation first. If no intersection is found, then use the color-matching interpolation instead to find the possible physical point using color consistency.

This strategy is justified by the following observations. Normally the intensity-based stereo vision algorithms are prone to error in weakly textured areas. This causes disparity maps generated to be rather noisy in these areas. Since a high degree of discontinuity exists, the threshold of zero-crossing required by the disparity-searching interpolation tends to be larger than the given threshold, and hence, no intersection is found. However, in the weakly textured areas, the color-matching interpolation can do a much better job. The result of the combined approach is shown in Figure 6-10(b). It shows that a smooth and detailed result can be generated.

6.3.3 Computational Cost Analysis

In the following, the computational costs needed for the above discussed interpolation approaches are discussed.

For each given ray, only the nearby four reference images are used for interpolation. This means that the computational cost is independent of the total number of reference images. As a result, both the color-matching and disparity-searching interpolations have a time complexity of $O(d \times r \times c)$, where r and c are the width and height of the image to be generated, and d is the difference between the minimum and maximum disparities of neighboring reference images.

Obviously, the more densely the rayset is sampled, the smaller is the value of d. When the dataset sampled by the light field is used, where d is close to 1, the speed of the algorithm is close to that of the light field rendering approach. With the value of d increasing, the cost of computations increases linearly, but the cost of sampling decreases quadratically. In addition, as the value of d increases, the directional sampling rate is reduced. This is a justified action when the scene consists of diffuse objects. However, when highly reflective surfaces exist in the scene, the highlights and reflections may not be reproduced very well. Therefore, it is also a tradeoff between data size and image

quality.

In practice, for the above "head and lamp" dataset, whose image resolution is 384×288 and maximum disparity is 14, the current implementation takes 0.2~0.4 second to render a frame (256×256 in size) on a 1.6GHz Pentium 4 PC running Windows 2000. Since the current implementation does not utilize any hardware acceleration and is not optimized, there is still room for improvements.

6.4 Experimental Results on Static Scenes

Before the algorithm is applied to dynamic scenes, it is evaluated using two static ones. The experiments conducted using these two scenes are given below.

6.4.1 Planar Camera Field Interpolation

The "Santa Claus" dataset, which is captured by Yuichi Ohta at the University of Tsukuba using a camera matrix, is used as a planar camera field. The original dataset contains 9×9 images with 640×480 resolution. To remove the black borders in the original images, the canvas size is reduced to 636×472.

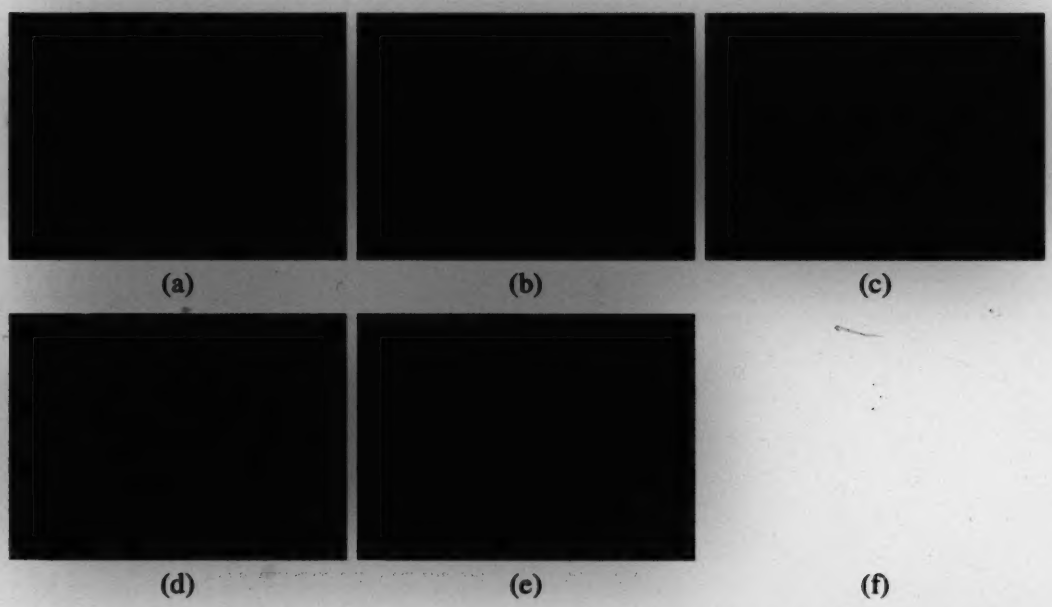


Figure 6-11: Interpolation results for the "Santa Claus" planar camera field.

Four reference images, whose coordinates are (3,3), (3,5), (5,3), and (5,5) in the

dataset, are used to interpolate the in-between view, i.e., the reference image, whose coordinates are (4,4) in the dataset. As a result, this reference image, shown in Figure 6-11(a), can be used as a ground truth to evaluate the rendering results. Since the baseline between neighboring reference image is very large (102 pixels, nearly one sixth of the image width), the linear interpolation method, as shown in Figure 6-11(b), gives a very blurry result. The result generated by the color-matching interpolation, shown in Figure 6-11(c), is sharp and smooth. However, closer inspection shows artifacts in areas around the mouth and the right eye. In addition, details on the hat of the Santa Claus and on the wall are lost. Figure 6-11(d) shows the result of the disparity-searching interpolation, by which the artifacts are removed and the details are preserved. The green pixels in Figure 6-11(d) are areas where no intersections are found. In the combined approach, shown in Figure 6-11(e), these areas are filled in by the color-matching interpolation. The absolute difference between the result of the combined approach and the ground truth is shown in Figure 6-11(f). The darker the area the larger is the error. One can see that the error is quite small.

6.4.2 Cylindrical Camera Field Interpolation

Since no cylindrical camera field for real scenes is available, an architectural model, which contains many fine details, is used to generate a cylindrical camera field. 72×10 images are used to sample the cylinder, resulting one reference image per 5 degrees in the horizontal direction. This is a very sparse sampling scheme as compared with the light field approach, in which 32×32 images are used to sample a 45×45 degree area of the "Buddha" model, and with the concentric mosaics, which sample 20 concentric circles in 3000 angular directions, resulting in one sample per 0.12 degrees in the horizontal direction.

Figure 6-12 shows the results of the algorithm using the synthetic camera field. The disparity information of the scene is obtained during the rendering process, and therefore, is accurate. Figure 6-12(a) shows the in-between view interpolated using linear interpolation. The result of the color-matching interpolation is shown in Figure 6-12(b). Artifacts show up near the boundary, and details, such as the frames for the windows and textures on the lawns, are lost. Figure 6-12(c) shows the "rubber sheet" artifacts of the

disparity-searching interpolation when no threshold is used. As shown in Figure 6-12(d), these artifacts are removed after the threshold, t=0.1, is used.

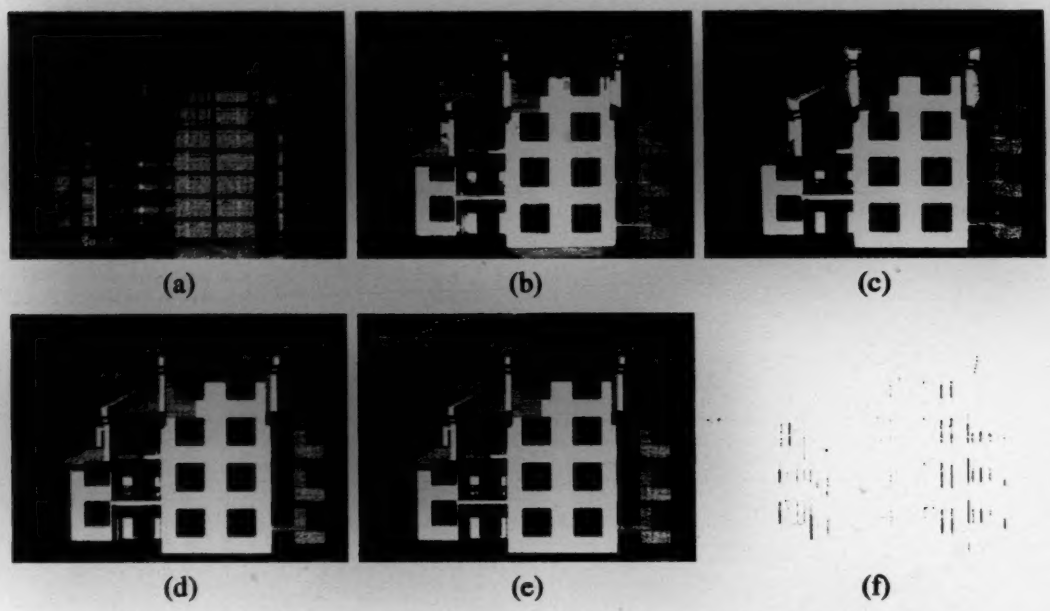


Figure 6-12: Interpolation results for a cylindrical camera field.

For comparison, the geometry-based rendering result for the in-between viewing position is also computed and is used as the ground truth, shown in Figure 6-12(e). The absolute difference between the camera field rendering result and the ground truth is shown in Figure 6-12(f). Even though Figure 6-12(f) shows large errors around the edges of surfaces, closer inspection indicates that these fine details do show up in Figure 6-12(d). The errors are actually caused by the interpolation, which tends to blend the edges.

6.4.3 Comparison with Dynamic Reparameterized Rendering

The above experiments show that, when the scene is sparsely sampled and there is no geometric information available, the rendering results will be very blurry. As it is reviewed before, the dynamic reparameterized rendering approach has tried to solve this problem by introducing a focal plane as additional geometric information. Here, an experiment is conducted to compare this approach with the camera field rendering.

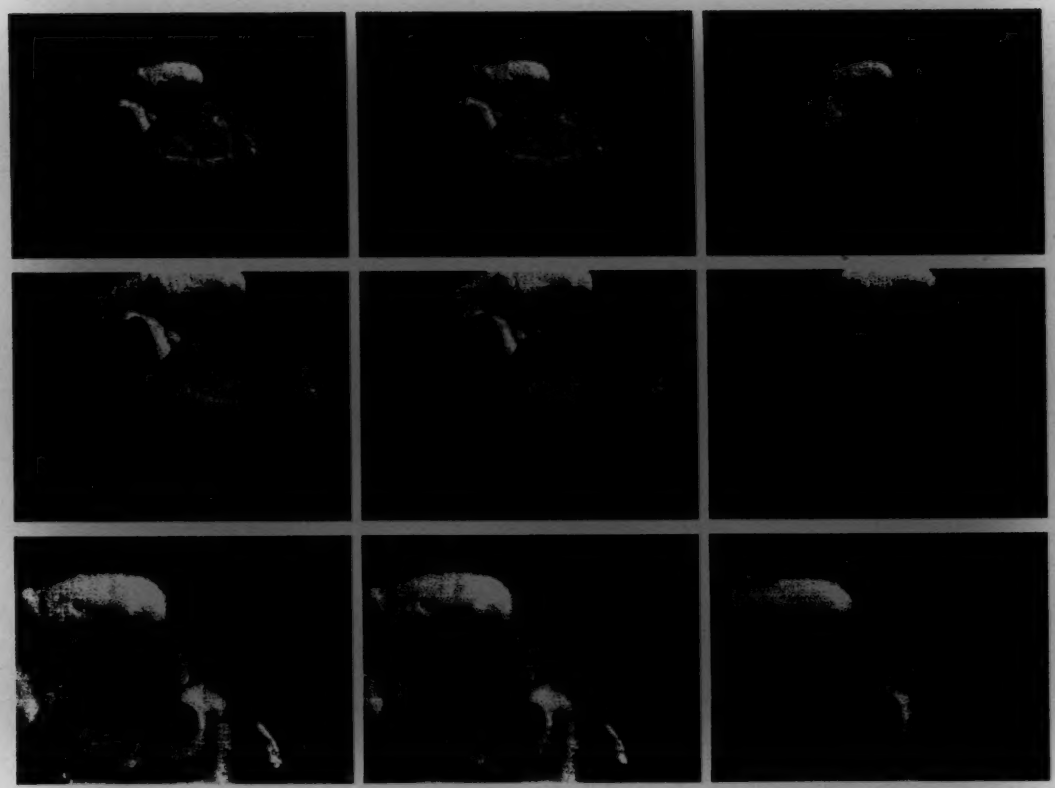


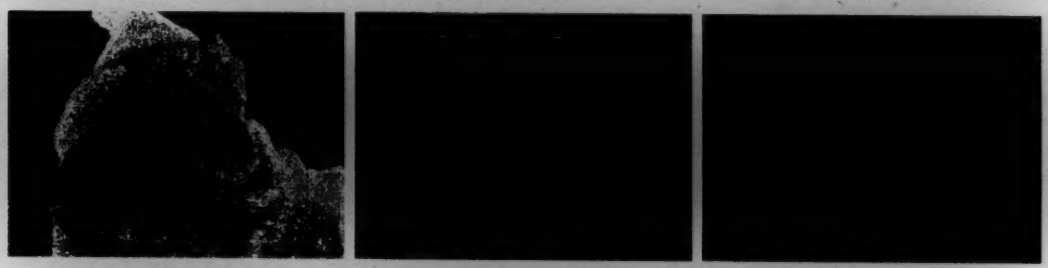
Figure 6-13: Comparison with dynamically reparameterized light field rendering.

In this experiment, all 9×9 reference images from the "Santa Claus" dataset are used to render arbitrary novel views. The rendering results for three camera positions are shown in Figure 6-13. In the figure, images in the first column are generated using the camera field rendering approach, and the rest are generated by the dynamic reparameterized rendering approach. The focal plane is set at the center of the Santa Claus for images in the second column, and is set at the location of the wall for those in the last column.

The results show that the camera field rendering approach can produce sharp rendering results for the whole scene in all these views. Since only sparsely sampled dataset is available, the rendering results generated by the dynamic reparameterized rendering approach have very limited depth of field. When the focal plane is set at the location of the wall, the Santa Claus is completely out of focus. Even when the focal plane is set at the center of the Santa Claus, where the eyes are, other parts of the Santa Claus, such as the hands and the boundary of the hat, are still out of the focus.

6.4.4 Robustness Evaluation

Finally, the robustness of the algorithm is evaluated using inaccurate disparity maps as input. Here, the Stereo by Eye Array approach is used to generate disparity maps for all reference images.



(a) (b) (c)

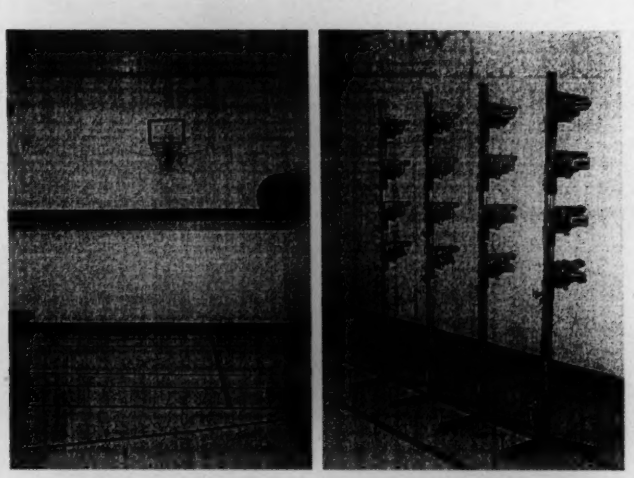
Figure 6-14: Rendering results using relative noisy disparity maps.

Figure 6-14(a) shows the disparity map for one of the reference images. It contains more errors than to the one generated by the reliability-based algorithm, shown in Figure 5-8(f). As shown in Figure 6-14(b), the disparity-searching interpolation can detect most of the errors. The pixels affected, shown in green color, are left unrendered and filled using the color-matching interpolation. As a result, the combined approach, shown in Figure 6-14(c), produces reasonably good results. This experiment suggests that the camera field rendering approach is not very sensitive to noise in the disparity maps provided.

On the other hand, it is noteworthy that the relatively accurate disparity maps provided by the reliability-based algorithm do help to remove errors in the rendering result. Comparison between Figure 6-14(c) and Figure 6-11(e) shows that many artifacts around the hat and the dress of the Santa Claus are removed when accurate disparity maps are used.

6.5 Experimental Results on a Dynamic Scene

In this experiment, 16 Sony TRV120 digital camcorders are used to record a dynamic scene on tapes. The DV formatted videos are later transferred into a computer offline through the FireWire interface.



(a) (b)

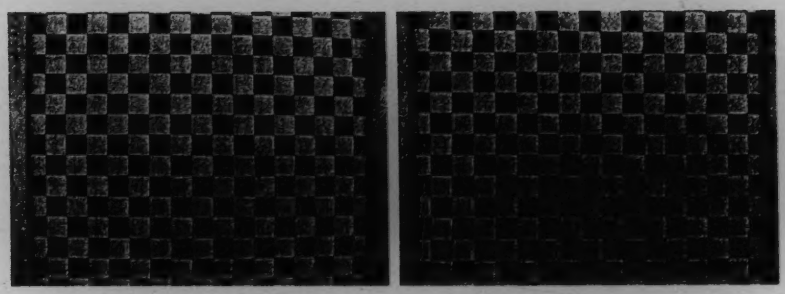
Figure 6-15: Experiment setup for dynamic IBR.

The dynamic scene captured is a ping pong action in a school gymnasium in Saskatoon. The scene, as shown in Figure 6-15(a) contains patterns on both the floor and the wall, which are later used for image rectification. As shown in Figure 6-15(b), sixteen camcorders are mounted on four custom made camera stands. The positions of the stands are equally spaced manually, and so are the positions of the camcorders on the stands. As a result, the centers of projections for different camcorders are equally spaced in both horizontal and vertical directions, even though the horizontal and the vertical baseline may not be the same. The camcorders are controlled by a single remote control, through which one can change the zoom lens and switch the recording on and off.

6.5.1 Camera Calibration

The camcorders are calibrated before the experiment. The calibration approach proposed by Zhang [191] is applied, which can give both the intrinsic and the radial distortion parameters. To use this approach, a checkerboard pattern is printed on a cardboard using a plotter. Images of the pattern under different viewing directions are taken using the camcorders. The corners in the pattern are detected, and the software provided by Zhang [192] is then used to calculate all the parameters.

The calibration results show that the parameters for different camcorders are similar. Therefore, a single set of parameters are used for all the camcorders in the experiment. The values of these parameters are listed in Appendix B.



(a) (b)

Figure 6-16: Radial distortion correction.

Figure 6-16(a) shows the checkerboard pattern captured by one of the camcorders. A slight barrel distortion is visible in the figure. The distortion is correctly removed in Figure 6-16(b) using the estimated parameters.

6.5.2 Video Synchronization

Through the experiment, it is found that the remote control can make different camcorders start recoding at roughly the same time, but not accurate enough for this work. To solve this problem, the captured videos are synchronized manually during the de-interlacing process.

The camcorders used capture the scene into NTSC videos. A NTSC video has 30 frames per second[1]. A frame is divided into two fields, each of which contains every other horizontal line in the frame. The field that contains the topmost scan line in the frame is called the upper field, and the other one is called the lower field. Since the two fields are captured at different times, horizontal streaks will show up around the boundaries of fast moving objects when the whole frame is displayed. To completely remove the interlacing effect, one of the two fields is discarded.

In the manual synchronize process, a synchronization event is chosen first. The field that contains a frame closer in time to the synchronization event is kept, while the other is thrown away. Since there are 60 fields per second, this manual synchronize process ensures that the synchronization error is less than $1/60^{th}$ seconds.

The resolution of the original DV video is 720×480. After de-interlacing, the

---

[1] To be precise, the frame rate of NTSC standard is $30\times1000\div1001\approx29.97$.

resolution drops to 720×240. In order to provide correct aspect ratios, the videos are resized to 320×240. The radial distortion correction is then applied to the resized images. Please note that the distortion correction cannot be performed before the de-interlacing step since, otherwise, it will mix pixels from different fields together.

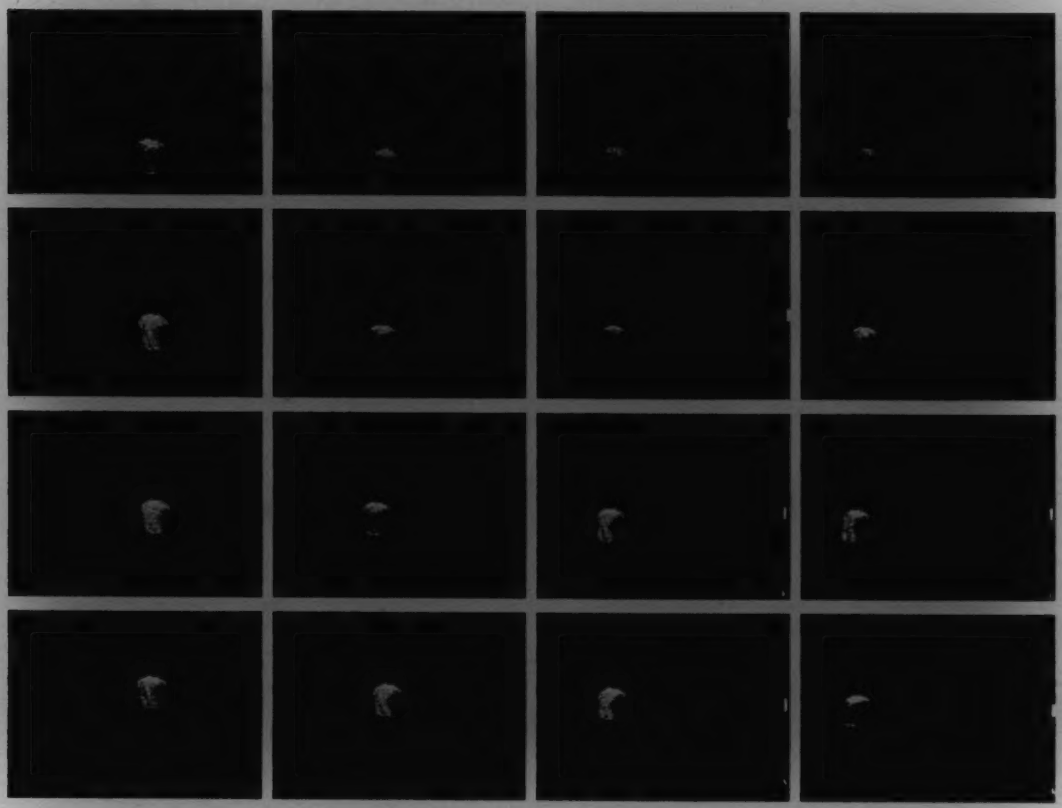


Figure 6-17: Synchronized frames in captured videos.

The synchronization event used in this experiment is that the ping pong ball touches the bat of the player in dark cyan t-shirt. Figure 6-17 shows the fields (after resizing) used for different videos that are that closer to this event. The figure also indicates that views observed by different cameras differ a lot.

6.5.3 Image Rectification

Using the camera mounting stands, it is possible to keep the centers of projections for different views equally spaced. However, the viewing directions of different camcorders may not be parallel to each other. Therefore, in order to satisfy the constraints for both the planar camera field rendering approach and the reliability-based stereo algorithm,

image rectification is needed to adjust the viewing directions.

Image rectification has been extensively investigated in computer vision. However, most of the proposed approaches study how to align two or three views [7, 54]. Therefore, they cannot be directly applied to adjust the view directions for all sixteen views, whose centers of projections are assumed to be lined up already. In this experiment, a rectification process based on vanishing points is applied.

(a) (b)

Figure 6-18: Image rectification using vanishing points.

Figure 6-18(a) shows the scene captured by one of the camcorders after radial distortion correction. The response after applying the Sobel operator is shown in Figure 6-18(b). The line features used are also shown in Figure 6-18(b) as green and red lines. These features form two sets, each of which consists of parallel lines. The vanishing point for each set is estimated using the least squares technique. Obviously, vanishing points close to infinity cannot be expressed in the image plane very well. Therefore, they are expressed as points on the Gaussian sphere, which is the unit sphere centered at the center of projection of the camera [8].

For a given reference view, assume that the center of the projection is C, the vanishing points calculated using the red parallel lines is V_0, and that calculated using the green parallel lines is V_1. The new coordinate system is defined such that the z-axis points to V_0 and the x-axis is on the plane CV_0V_1. The rotation matrix needed for this reference view can then be calculated. Since the vanishing points are invariants, after rotation, the view directions for different reference views are parallel to each other.

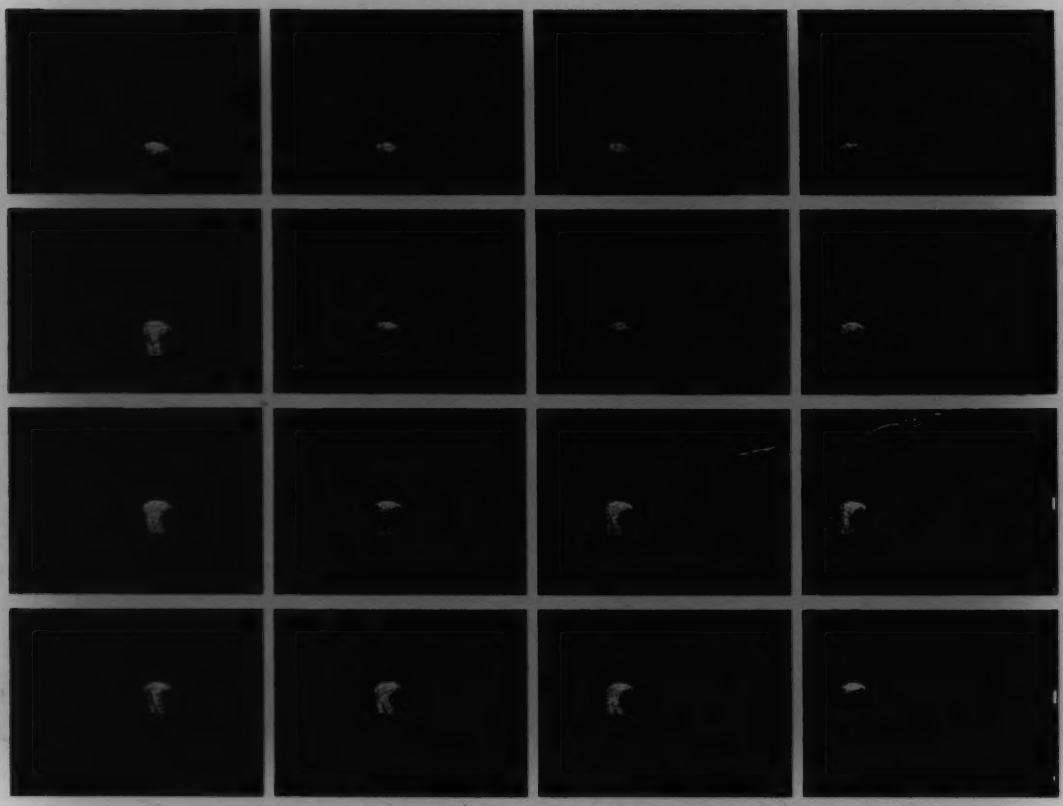


Figure 6-19: Different views after rectification.

The synchronization frames from different videos after rectification are shown in Figure 6-19. In the image, green denotes pixels that are mapped to outside of the original views.

6.5.4 Disparity Pre-calculation

Compared with the "head and lamp" and "Santa Claus" datasets, which are captured by the camera matrix, the ping pong dataset is much more challenging for accurate stereo matching. This is due to the following reasons:

- Images captured by the camcorder are much noisier due to both CCD sensor quality and DV compression applied.
- The white balance for different camcorders is different, resulting in the colors of the same object to differ a lot in different videos.
- The videos are not exactly synchronized, causing the 3D positions of fast moving objects captured by different camcorders to be different.

- The camera stands may not align all the centers of projections exactly such that they are equally spaced and on the same plane.

(a) (b) (c)

Figure 6-20: Disparity maps calculated using different algorithms.

For example, Figure 6-20(a) shows the result generated for one of the center views using the Stereo by Eye Array approach. Many errors appear in this disparity map and, due to a low signal-to-noise ratio, the player in black t-shirt is not distinguishable with the background. As shown in Figure 6-20(b), the result generated by the reliability-based algorithm is more promising, even through solutions are not provided in some areas (shown in white) due to low reliability. These areas could be filled using heuristic approaches, such as the median filter used in Chapter 5. However, as shown in Figure 6-20(c), simple heuristic filling will introduce additional errors, such as in the area around the rightmost person's head.

A better way to solve the problem is to leave the disparities for these areas unassigned. The result is that the disparity-searching interpolation will not be able to find intersections for certain areas due to missing disparity information. However, the color-matching interpolation is unaffected. Therefore, the rendering results generated by the combined approach are likely to be better than using complete but incorrect disparity maps.

Figure 6-21 shows the disparity maps calculated for different views. It is noteworthy that the disparity maps at the boundary have more low reliability areas than those in the center. This is because only two or three neighboring views are available when calculating boundary disparity maps, while four neighboring views are available when calculating the center disparity maps.

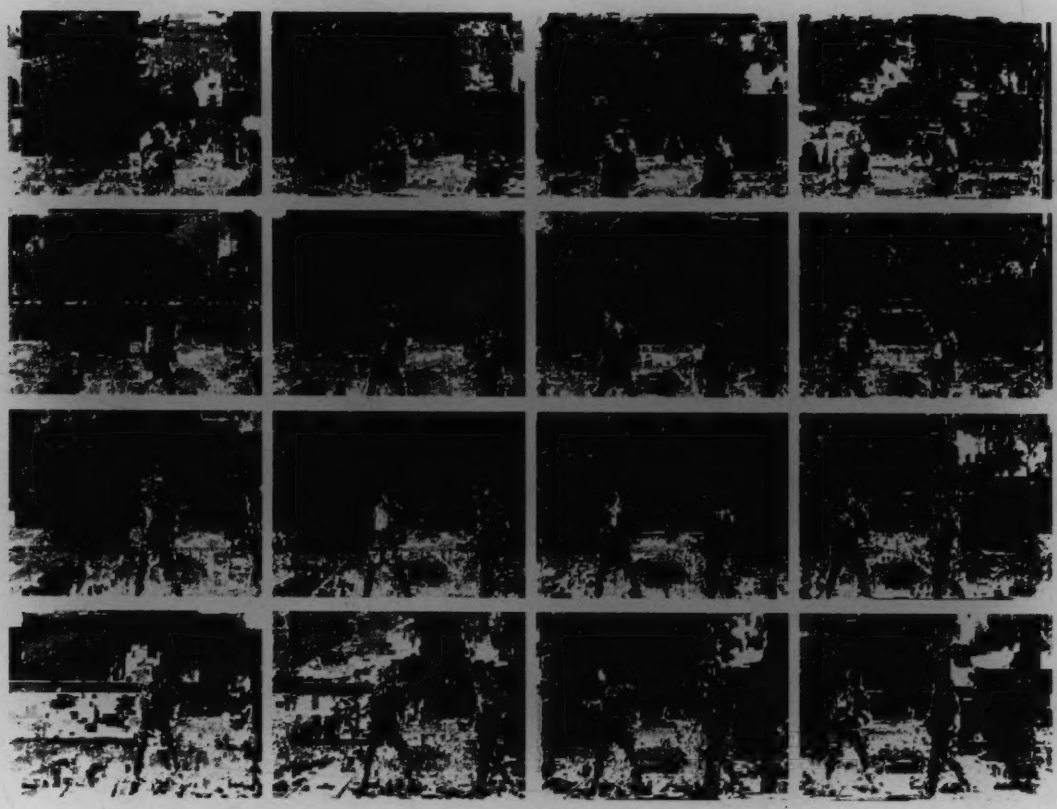


Figure 6-21: Disparity maps calculated for different views.

6.5.5 Rendering Results

In this subsection, the rendering result for the synchronized frame is shown first, followed by the results at different time stamps.

Figure 6-22 shows the results generated through interpolating the four views in the center. Figure 6-22(a) is generated using linear interpolation, which shows that the nearby four views differ a lot. Figure 6-22(b) and (c) are generated using dynamic reparameterized rendering technique by focusing on the player in the white and the black t-shirt, respectively. The results indicate that due to the sparse sampling rate used, the depth of field is very limited. Figure 6-22(d) and (e) are generated by the color-matching and the disparity-searching interpolations, respectively. It is noteworthy that some areas in Figure 6-22(e) are left unrendered due to the missing information in the input disparity maps. Finally, Figure 6-22(f) shows the result of combined approach, which gives reasonably good results on details such as patterns on the floor and textures on the wall.

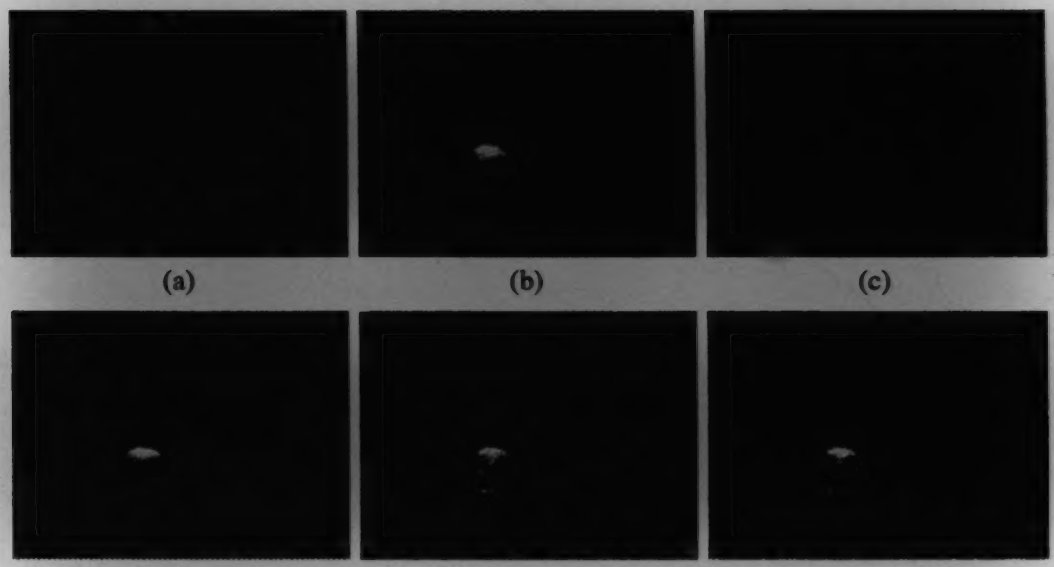


Figure 6-22: Interpolation results using different approaches.

The rendering results for different time stamps are shown in Figure 6-23. The results show that the algorithm is able to correctly handle occlusions when the player in dark cyan t-shirt walks in front of the other two.

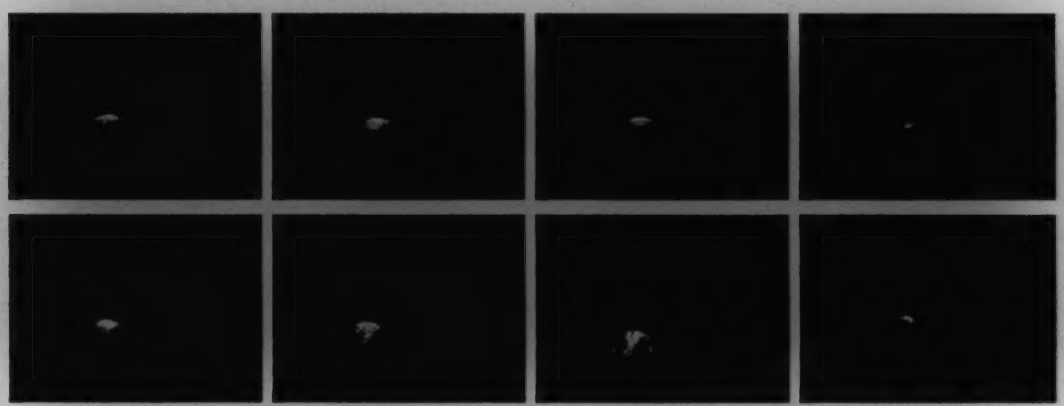


Figure 6-23: Different frames in the generated video.

Chapter 7
Conclusion and Future Work

Image-based modeling and rendering is an exciting and challenging topic, which links the domains of traditional computer graphics and computer vision. For instance, it involves both scene reconstruction, which is a vision problem, and novel view generation, which is a graphics problem.

This dissertation investigates problems in the IBR area from the graphics as well as from the vision perspective. On the computer graphics side, several novel IBR techniques are developed, including one approach that is suitable for handling dynamic scenes. On the computer vision side, a novel stereo vision algorithm is presented, which can generate reasonably accurate disparity maps even for challenging datasets.

In the rest of this chapter, the contributions of this dissertation are discussed first, followed by limitations and possible future works.

7.1 Contributions

The main contributions of this research include: the rayset taxonomy, the reliability-based stereo vision algorithm, and the dynamic scene rendering technique.

7.1.1 Rayset Taxonomy

First of all, the rayset is a new concept. Comparing with images, the rayset concept explicitly defines the mapping relation between the parameter space and the ray space. Conventional approaches either ignore this mapping relation or define it implicitly. In the

latter case, the mapping relation is usually specified using internal and external parameters of the camera. Furthermore, raysets can be defined in a high-dimensional parameter space, which is another extension to conventional 2D images.

Comparing with the plenoptic model, the rayset concept separates the mapping from the ray space to the attribute space into two separate mapping relations. The first mapping relation defines how the ray space is sampled, while the second defines the sampling result. Such a separation introduces flexibilities. For example, changing the support function without altering the attribute function can produce the effect of distorting the scene. On the other hand, keeping the support function unaffected and changing the attribute function gives the effect of changing the appearance of objects in the scene.

The rayset is also a taxonomy that can be used to classify existing IBR techniques. Under the rayset taxonomy, the scene representation technique is separated from the scene reconstruction technique. With respect to scene representations, they are classified into planar images, non-planar images, and high-dimensional representations. Similarly, scene reconstruction techniques are classified into plenoptic sampling, depth warping, texture mapping, and scene morphing. The above classifications demonstrate the relations among different approaches. Therefore, the rayset taxonomy is a useful tool in understanding of techniques developed in the IBR area as well as in developing new techniques.

7.1.2 Novel Stereo Vision Algorithm

Depth or disparity information is very helpful in reducing the sampling density requirement. To estimate accurate disparity information based on different views, a novel stereo vision algorithm is designed.

Most previous stereo vision approaches rely on the smoothness constraint to remove mismatches, which will also remove some details. To address this problem, a new reliability measure is introduced. The measure is defined based on the cost difference between the best alternate path and the path under use and can therefore be employed in general dynamic programming approaches. Using the reliability measure to remove mismatches gives better results than using the smoothness constraint since fine details with enough reliability can be preserved.

The strong and the weak consistency constraints are also defined by re-formulating and extending the commonly used consistency check. These constraints are used to filter out potential mismatches. Since the weak consistency constraint explicitly models the visibility in the image space, it can be applied to both occluded areas and areas that contain thin foreground objects.

When integrating the reliability thresholding process and the consistency constraints through relaxation, one can choose to increase the discontinuity cost gradually. Consequently, matches of the most distinct features are confirmed first. Those in noisy and textureless areas will not be accepted until there are enough supports from neighboring matches. This gives the effects of automatically adjusting the smoothness weight at different regions of the image. Basically, the higher the signal-to-noise ratio of a region, the sooner the matches in the region will be confirmed, resulting in a smaller discontinuity cost to be applied in the region. As a result, smooth and detailed disparity maps can be generated.

7.1.3 Dynamic Scene Rendering Technique

While different views of static scenes can be captured by moving a single camera, the views of dynamic scenes must be captured by separate physical cameras. The number of views is limited by the number of cameras available, and more importantly, the sampling density is limited by the size of the cameras used. As a result, dynamic IBR techniques must deal with sparsely sampled datasets. A plenoptic sampling based approach for a real dynamic scene is not practical.

To address the issue of blurry results produced by simple interpolation when only sparse samples are available, two novel interpolation techniques are presented. The first one, the color-matching interpolation, does not require explicit depth information. For any given testing ray, it searches for a possible physical point along the testing ray using color information of nearby reference images. If more than one point is found, the one closest to the center of projection is used. This approach is robust, but it may give incorrect interpolation result when two assumptions are not satisfied.

The second technique, the disparity-searching interpolation, employs the calculated disparity maps for the reference images. A backward searching process is used to find the

closest intersection between the testing ray and the disparity surface defined by the nearby reference images. This approach can generate correct results even when occlusions exist. However, it may give holes when the calculated disparity maps are noisy.

The two interpolation techniques are combined together to provide a robust rendering algorithm is to combine. That is, try to search for an intersection using the disparity-searching interpolation first. If no intersection is found, then the color-matching interpolation is used to fill the hole. This strategy works very well since most intensity-based stereo vision algorithms are prone to error in textureless areas, the area that the color-matching interpolation can do a very good job.

Both the color-matching and disparity-searching interpolations are backward rendering techniques. Comparing with the forward rendering techniques [154, 155], they have the advantages that there is no need to reproject all the samples and to fill the Z-buffer.

7.2 Future Work

There is much future work to be done, mostly in the dynamic IBR area. In this last section, two major possible future research directions are discussed.

7.2.1 Stereo-Motion Analysis

As shown in Section 6.5, when handling dynamic scenes, the geometric information is estimated using the reliability-based algorithm discussed in Chapter 5. The disparity map calculation is done in a per frame basis, and therefore, the temporal coherence between adjacent frames is ignored. Obviously, this coherence can be employed to remove ambiguities and to detect potential mismatches.

The temporal coherence is studied in the computer vision area as the motion estimation problem, which is similar to the problem of the stereo vision since both are matching problems. However, instead of matching among images taken at different locations, motion estimation matches among images taken at different time. For dynamic IBR, since multiple video sequences are available, it is possible to combine stereo and motion techniques. Such a fusion, i.e., the stereo-motion analysis, makes it possible to

estimate the 3D motion of objects in the scene.

The idea of using the reliability measure in disparity calculation can be extended to stereo-motion analysis as well. Currently, disparity maps for different views are calculated simultaneously and the consistency constraints are used to detect inconsistency between disparities assigned to different views. It is also feasible to match between images taken at adjacent time stamps and compare the disparities assigned to pixels involved in the match. If the two disparities differ a lot, it is highly possible that one or both pixels have incorrect disparity assigned.

7.2.2 Unstructured Camera Rendering

One major limitation of the camera field rendering approach is that it requires the images be captured using a pre-defined structure. Hence, the cameras must be aligned on a plane or on a parametric surface, such as a cylinder. This makes the setup of the system somewhat difficult.

The concepts of the two interpolations approaches using in the camera field rendering approach can be applied to arbitrary camera setting as long as the positions and orientations of the cameras are known. Hence, it is theoretically feasible to extend the camera field rendering to unstructured views. Assume that the cameras are fully calibrated, the epipolar line can be determined first, and then the suitable pixel along the epipolar line can be located. What needs to be done is to derive the distance function, E, and intersection searching function, F, for the general case.

The temporal coherence that exists in video sequences can also be utilized in the rendering process. In camera field rendering, different frames in the videos are rendered independently. Obviously, less computation is required if only the moving parts of the scene are re-rendered. Such an acceleration approach will be practical if the stereo-motion analysis approach can provide accurate information about the locations and movements of moving objects in the scene.

References

[1] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, M. S. Landy and A. J. Movshon, Eds., Cambridge, MA, USA: The MIT Press, 1991.

[2] D. G. Aliaga, "Visualization of complex models using dynamic texture-based simplification," in *Proc. IEEE Visualization*, pp. 101-106, San Francisco, CA, USA, October 27-November 1, 1996.

[3] D. G. Aliaga and A. A. Lastra, "Architectural walkthroughs using portal textures," in *Proc. IEEE Visualization*, pp. 355-362, Phoenix, AZ, USA, October 19-24, 1997.

[4] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855-867, September 1990.

[5] K.-i. Anjyo, Y. Usami, and T. Kurihara, "Simple method for extracting the natural beauty of hair," in *Proc. Siggraph Annual Conference*, pp. 111-120, Chicago, IL, USA, July 26-31, 1992.

[6] M. Aono, "Attribute mapping: Concept and implementation," *Computers and Graphics*, vol. 14, no. 3-4, pp. 465-475, 1990.

[7] N. Ayache and C. Hansen, "Rectification of images for binocular and trinocular stereovision," in *Proc. International Conference on Pattern Recognition, vol. 1*, pp. November 14-17, Rome, Italy, October, 1988.

[8] S. T. Barnard, "Interpreting perspective images," *Artificial Intelligence*, vol. 21, no. 4, pp. 435-462, November 1983.

[9] S. T. Barnard, "Stochastic stereo matching over scale," *International Journal of Computer Vision*, vol. 3, no. 1, pp. 17-32, May 1989.

[10] J. Battle, E. Mouaddib, and J. Salvi, "Recent progress in coded structured light as a technique to solve the correspondence problem: A survey," *Pattern Recognition*, vol. 31, no. 7, pp. 963-982, July 1998.

[11] T. Beier and S. Neely, "Feature-based image metamorphosis," in *Proc. Siggraph Annual Conference*, pp. 35-42, Chicago, IL, USA, July 26-31, 1992.

[12] V. Blanz and T. Vetter, "Morphable model for the synthesis of 3D faces," in *Proc. Siggraph Annual Conference*, pp. 187-194, Los Angeles, CA, USA, August 8-13, 1999.

[13] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *Proc. Siggraph Annual Conference*, pp. 192-198, San Jose, CA, USA, July 20-22, 1977.

[14] J. F. Blinn, "Simulation of wrinkled surfaces," in *Proc. Siggraph Annual Conference*, pp. 286-292, New York, NY, USA, August 23-25, 1978.

[15] J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Communications of the ACM*, vol. 19, no. 10, pp. 542-547, October 1976.

[16] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *International Journal of Computer Vision*, vol. 33, no. 3, pp. 181-200, September 1999.

[17] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222-1239, 2001.

[18] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," in *Proc. Siggraph Annual Conference*, pp. 425-432, Los Angeles, August 12-17, 2001.

[19] P. J. Burt, "Moment images, polynomial fit filters, and the problem of surface interpolation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 144-152, Ann Arbor, MI, USA, June 5-9, 1988.

[20] E. Camahort, A. Lerios, and D. Fussell, "Uniformly sampled light fields," in *Proc. Eurographics Workshop on Rendering*, pp. 117-130, Vienna, Austria, June 29-July 1, 1998.

[21] E. Catmull, "A subdivision algorithm for computer display of curved surface,"

Ph.D. Dissertation, University of Utah, 1974.

[22] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," in *Proc. Siggraph Annual Conference*, pp. 307-318, New Orleans, LA, USA, July 23-28, 2000.

[23] C.-F. Chang, G. Bishop, and A. A. Lastra, "LDI tree: A hierarchical representation for image-based rendering," in *Proc. Siggraph Annual Conference*, pp. 291-298, Los Angeles, CA, USA, August 8-13, 1999.

[24] D. T. Chen and D. Zeltzer, "Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method," in *Proc. Siggraph Annual Conference*, pp. 89-98, Chicago, IL, USA, July 26-31, 1992.

[25] Q. Chen and G. Medioni, "Volumetric stereo matching method: Application to image-based modeling," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition vol. 1*, pp. 29-34, Fort Collins, CO, USA, June 23-25, 1999.

[26] S. E. Chen, "Quicktime VR - an image-based approach to virtual environment navigation," in *Proc. Siggraph Annual Conference*, pp. 29-38, Los Angeles, CA, USA, August 9-11, 1995.

[27] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *Proc. Siggraph Annual Conference*, pp. 279-285, Anaheim, CA, USA, August 1-6, 1993.

[28] C. H. Chien and J. K. Aggarwal, "Volume/surface octrees for the representation of three-dimensional objects," *Computer Vision, Graphics, and Image Processing*, vol. 36, no. 1, pp. 100-113, October 1986.

[29] C. H. Chien, Y. B. Sim, and J. K. Aggarwal, "Generation of volume/surface octree from range data," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 254-260, Ann Arbor, MI, USA, June 5-9, 1988.

[30] D. Cobzas, K. Yerex, and M. Jagersand, "Dynamic textures for image-based rendering of fine-scale 3D structure and animation of non-rigid motion," in *Proc. Eurographics Annual Conference*, Saarbrucken, Germany, September 9-10, 2002.

[31] D. Cohen and A. Shaked, "Photo-realistic imaging of digital terrains," in *Proc. Eurographics Annual Conference*, pp. 363-373, Barcelona, Spain, 1993.

[32] M. F. Cohen, M. Levoy, J. Malik, L. McMillan, and S. E. Chen, "Image-based rendering: Really new or deja vu?" in *Proc. Siggraph Annual Conference*, pp. 468-470, Los Angeles, CA, USA, August 3-8, 1997.

[33] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 358-363, San Francisco, CA, USA, June 18-20, 1996.

[34] R. L. Cook, "Shade trees," in *Proc. Siggraph Annual Conference*, pp. 223-231, Minneapolis, MN, USA, July 23-27, 1984.

[35] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs, "A maximum likelihood stereo algorithm," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 542-567, May 1996.

[36] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. Siggraph Annual Conference*, pp. 207-212, Minneapolis, MN, USA, July 23-27, 1984.

[37] B. W. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized voxel coloring," in *Proc. ICCV Workshop on Vision Algorithms: Theory and Practice*, pp. 100-115, Kerkyra, Corfu, Greece, September 21-22, 1999.

[38] B. Curless and M. Levoy, "Volumetric method for building complex models from range images," in *Proc. Siggraph Annual Conference*, pp. 303-312, New Orleans, LA, USA, August 4-9, 1996.

[39] W. J. Dally, L. McMillan, G. Bishop, and H. Fuchs, *The delta tree: An object-centered approach to image-based rendering*, Technical Memo 1604, AI Lab, Massachusetts Institute of Technology, May 1996.

[40] P. E. Debevec, "The Campanile movie and The Matrix," http://www.debevec.org/Campanile/, 2001. (last accessed June 1, 2003)

[41] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *Proc. Siggraph Annual Conference*, pp. 11-20, New Orleans, LA, USA, August 4-9, 1996.

[42] P. E. Debevec, Y. Yu, and G. Borshukov, "Efficient view-dependent image-based rendering with projective texture-mapping," in *Proc. Eurographics Workshop on Rendering*, pp. 105-116, Vienna, Austria, June 29-July 1, 1998.

[43] X. Decoret, G. Schaufler, F. Sillion, and J. Dorsey, "Multi-layered impostors for accelerated rendering," in *Proc. Eurographics Annual Conference*, pp. C61-C72, Milano, Italy, September 7-11, 1999.

[44] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *Proc. Siggraph Annual Conference*, pp. 275-286, Orlando, FL, USA, July 19-24, 1998.

[45] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. K. Pedersen, "Modeling and rendering of weathered stone," in *Proc. Siggraph Annual Conference*, pp. 225-234, Los Angeles, CA, USA, August 8-13, 1999.

[46] T. A. Funkhouser and C. H. Sequin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments," in *Proc. Siggraph Annual Conference*, pp. 247-254, Anaheim, CA, USA, August 1-6, 1993.

[47] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 858-863, Puerto Rico, June 17-19, 1997.

[48] P. Gao and T. W. Sederberg, "Work minimization approach to image morphing," *Visual Computer*, vol. 14, no. 8-9, pp. 390-400, 1998.

[49] G. Y. Gardner, "Simulation of natural scenes using textured quadric surfaces," in *Proc. Siggraph Annual Conference*, pp. 11-20, Minneapolis, MN, USA, July 23-27, 1984.

[50] D. Geiger, B. Ladendorf, and A. Yuille, "Occlusions and binocular stereo," in *Proc. European Conference on Computer Vision*, pp. 425-433, Santa Margherita Ligure, Italy, 1992.

[51] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721-741, November 1984.

[52] C. A. Glasbey and K. V. Mardia, "A review of image warping methods," *Journal of Applied Statistics*, vol. 25, no. 2, pp. 155-171, 1998.

[53] A. Glassner, "Adaptive precision in texture mapping," in *Proc. Siggraph Annual Conference*, pp. 297-306, Dallas, TX, USA, August 18-22, 1986.

[54] J. M. Gluckman and S. K. Nayar, "Rectifying transformations that minimize resampling effects," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, USA, 2001.

[55] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," in *Proc. Siggraph Annual Conference*, pp. 213-222, Minneapolis, MN, USA, July 23-27, 1984.

[56] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proc. Siggraph Annual Conference*, pp. 43-54, New Orleans, LA, USA, August 4-9, 1996.

[57] N. Greene, "Environment mapping and other applications of world projection," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 21-29, November 1986.

[58] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society, Series B*, vol. 51, no. 2, pp. 271-279, 1989.

[59] W. E. L. Grimson, "Computational experiments with a feature based stereo algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 1, pp. 17-34, January 1985.

[60] J. P. Grossman and W. J. Dally, "Point sample rendering," in *Proc. Eurographics Workshop on Rendering*, pp. 181-192, Vienna, Austria, June 29-July 1, 1998.

[61] P. Haeberli and M. Segal, "Texture mapping as a fundamental drawing primitive," in *Proc. Eurographics Workshop on Rendering*, pp. 259-266, Paris, France, June 14-16, 1993.

[62] P. S. Heckbert, "Survey of texture mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56-67, November 1986.

[63] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. J. Van Gool, "Plenoptic modeling and rendering from image sequences taken by a hand-held camera," in *Proc. Symposium fur Mustererkennung*, pp. 94-101, Bonn, Germany, September 15-17, 1999.

[64] T. H. Hong and M. O. Shneier, "Describing a robot's workspace using a sequence of views from a moving camera," *IEEE Transactions on Pattern Analysis and*

Machine Intelligence, vol. 7, no. 6, pp. 721-726, November 1985.

[65] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proc. Siggraph Annual Conference*, pp. 71-78, Chicago, IL, USA, July 26-31, 1992.

[66] Y. Horry, K.-i. Anjyo, and K. Arai, "Tour into the picture: Using a spidery mesh interface to make animation from a single image," in *Proc. Siggraph Annual Conference*, pp. 225-232, Los Angeles, CA, USA, August 3-8, 1997.

[67] C. M. Huang and R. W. Harris, "Comparison of several vector quantization codebook generation approaches," *IEEE Transactions of Image Processing*, vol. 2, no. 1, pp. 108-112, January 1993.

[68] H.-C. Huang, S.-H. Nain, Y.-P. Hung, and T. Cheng, "Disparity-based view morphing - a new technique for image-based rendering," in *Proc. Symposium on Virtual Reality Software and Technology*, pp. 9-16, Taipei, Taiwan, November 2-5, 1998.

[69] T. Huang and A. Netravali, "Motion and structure from feature correspondences: A review," *Proceedings of the IEEE*, vol. 82, no. 2, pp. 252-267, 1994.

[70] I. Ihm, S. Park, and R. K. Lee, "Rendering of spherical light fields," in *Proc. Pacific Conference on Computer Graphics and Applications*, pp. 59-68, Seoul, South Korea, October, 1997.

[71] S. S. Intille and A. F. Bobick, "Disparity-space images and large occlusion stereo," in *Proc. European Conference on Computer Vision*, pp. 179-186, Stockholm, Sweden, May 2-6, 1994.

[72] A. Isaksen, L. McMillan, and S. J. Gortler, "Dynamically reparameterized light fields," in *Proc. Siggraph Annual Conference*, pp. 297-306, New Orleans, LA, USA, July 23-28, 2000.

[73] H. W. Jensen, "Global illumination using photon maps," in *Proc. Eurographics Workshop on Rendering*, pp. 21-30, New York City, NY, USA, June, 1996.

[74] J. T. Kajiya, "The rendering equation," in *Proc. Siggraph Annual Conference*, pp. 143-150, Dallas, TX, USA, August 18-22, 1986.

[75] T. Kanade, "Carnegie Mellon goes to Super Bowl," http://www.ri.cmu.edu/events/sb35/tksuperbowl.html, 2001. (last accessed June 1,

2003)

[76] T. Kanade and M. Okutomi, "Stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920-932, September 1994.

[77] K. Kanatani and T. Chou, "Shape from texture: General principle," *Artificial Intelligence*, vol. 38, no. 1, pp. 1-48, February 1989.

[78] S.-B. Kang, "A survey of image-based rendering techniques," in *Proc. International Symposium on Electronic Imaging: Videometrics VI vol. SPIE 3641*, pp. 2-16, San Jose, CA, USA, January 28-29, 1999.

[79] S.-B. Kang, R. Szeliski, and J.-X. Chai, "Handling occlusions in dense multi-view stereo," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 103-117, Kauai, Hawaii, USA, December 8-14, 2001.

[80] P. Kauff, O. Schreer, and R. Tanger, "Virtual team user environments: A mixed reality approach for immersive tele-collaboration," in *Proc. International Workshop on Immersive Telepresence*, Juan Les Pins, France, December 6, 2002.

[81] J. Kautz and H.-P. Seidel, "Hardware accelerated displacement mapping for image based rendering," in *Proc. Graphics Interface*, pp. 61-70, Ottawa, Ontario, Canada, June 7-9, 2001.

[82] T. J. Keating, P. R. Wolf, and F. L. Scarpace, "Improved method of digital image correlation," *Photogrammetric Engineering and Remote Sensing*, vol. 41, no. 8, pp. 993-1002, August 1975.

[83] C. Kolb, D. Mitchell, and P. Hanrahan, "Realistic camera model for computer graphics," in *Proc. Siggraph Annual Conference*, pp. 317-324, Los Angeles, CA, USA, August 9-11, 1995.

[84] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions via graph cuts," in *Proc. International Conference on Computer Vision vol. 2*, pp. 508-515, Vancouver, Canada, July 9-12, 2001.

[85] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *Proc. European Conference on Computer Vision*, pp. 82-96, Copenhagen, Denmark, May 27-June 2, 2002.

[86] K. N. Kutulakos, "Shape from the light field boundary," in *Proc. IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 53-59, San Juan, PR, USA, June 17-19, 1997.

[87] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," in *Proc. International Conference on Computer Vision*, pp. 307-314, Kerkyra, Corfu, Greece, September 20-25, 1999.

[88] A. Laurentini, "Visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 150-162, February 1994.

[89] S. Laveau and O. D. Faugeras, "3-D scene representation as a collection of images," in *Proc. International Conference on Pattern Recognition vol. 1*, pp. 689-691, Jerusalem, Israel, October 9-13, 1994.

[90] S.-Y. Lee, K.-Y. Chwa, J. Hahn, and S.-Y. Shin, "Image morphing using deformable surfaces," in *Proc. Computer Animation*, pp. 31-39, Geneva, Switzerland, 1994.

[91] S.-Y. Lee, K.-Y. Chwa, S.-Y. Shin, and G. Wolberg, "Image metamorphosis using snakes and free-form deformations," in *Proc. Siggraph Annual Conference*, pp. 439-448, Los Angeles, CA, USA, August 9-11, 1995.

[92] S.-Y. Lee, G. Wolberg, and S.-Y. Shin, "Polymorph: Morphing among multiple images," *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp. 58-71, January-February 1998.

[93] T.-Y. Lee, Y.-C. Lin, L. Lin, and Y. N. Sun, "Fast feature-based metamorphosis and operator design," in *Proc. Eurographics Annual Conference*, pp. 15-22, Lisbon, Portugal, August 31-September 4, 1998.

[94] J. Lengyel and J. Snyder, "Rendering with coherent layers," in *Proc. Siggraph Annual Conference*, pp. 233-242, Los Angeles, CA, USA, August 3-8, 1997.

[95] A. Lerios, C. D. Garfinkle, and M. Levoy, "Feature-based volume metamorphosis," in *Proc. Siggraph Annual Conference*, pp. 449-456, Los Angeles, CA, USA, August 9-11, 1995.

[96] M. D. Levine, D. A. O'Handley, and G. M. Yagi, "Computer determination of depth maps," *Computer Graphics and Image Processing*, vol. 2, no. 4, pp. 131-150, October 1973.

[97] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. Siggraph Annual Conference*, pp. 31-42, New Orleans, LA, USA, August 4-9, 1996.

[98] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3D scanning of large statues," in *Proc. Siggraph Annual Conference*, pp. 131-144, New Orleans, LA, USA, July 23-28, 2000.

[99] X. Li and M. Moshell, "Modeling soil: Real time dynamic models for soil slippage and manipulation," in *Proc. Siggraph Annual Conference*, pp. 361-368, Anaheim, CA, USA, August 1-6, 1993.

[100] A. Lippman, "Movie-maps: An application of the optical videodisc to computer graphics," in *Proc. Siggraph Annual Conference*, pp. 32-42, Seattle, WA, USA, July 14-18, 1980.

[101] D. Lischinski and A. Rappoport, "Image-based rendering for non-diffuse synthetic scenes," in *Proc. Eurographics Workshop on Rendering*, pp. 301-314, Vienna, Austria, June 29-July 1, 1998.

[102] P. Litwinowicz and L. Williams, "Animating images with drawings," in *Proc. Siggraph Annual Conference*, pp. 409-412, Orlando, FL, USA, July 24-29, 1994.

[103] Z. Liu, Y. Shan, and Z. Zhang, "Expressive expression mapping with ratio images," in *Proc. Siggraph Annual Conference*, pp. 271-276, Los Angeles, CA, USA, August 12-17, 2001.

[104] J. R. Logie and J. W. Patterson, "Inverse displacement mapping in the general case," *Computer Graphics Forum*, vol. 14, no. 5, pp. 261-273, December 1995.

[105] P. W. Maciel and S. Peter, "Visual navigation of large environments using textured clusters," in *Proc. Symposium on Interactive 3D Graphics*, pp. 95-102, Monterey, CA, USA, April 9-12, 1995.

[106] R. Manduchi and C. Tomasi, "Distinctiveness maps for image matching," in *Proc. International Conference on Image Analysis and Processing*, pp. 26-31, Venice, Italy, 1999.

[107] W. R. Mark, L. McMillan, and G. Bishop, "Post-rendering 3D warping," in *Proc. Symposium on Interactive 3D Graphics*, pp. 7-16, Providence, RI, USA, April 27-30, 1997.

[108] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, no. 4262, pp. 283-287, October 1976.

[109] J. Marroquin, S. Mitte, and T. Poggio, "Probabilistic solution for ill-posed problems in computational vision," *Journal of the American Statistical Association*, vol. 82, no. 397, pp. 76-89, March 1987.

[110] W. N. Martin and J. K. Aggarwal, "Volumetric descriptions of objects from multiple views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 150-158, March 1983.

[111] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-based visual hulls," in *Proc. Siggraph Annual Conference*, pp. 369-374, New Orleans, LA, USA, July 23-28, 2000.

[112] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan, "Image-based 3D photography using opacity hulls," in *Proc. Siggraph Annual Conference*, pp. 427-437, San Antonio, TX, USA, July 21-26, 2002.

[113] N. Max, "Hierarchical rendering of trees from precomputed multi-layer z-buffers," in *Proc. Eurographics Workshop on Rendering*, pp. 165-174, Porto, Portugal, June, 1996.

[114] N. Max and K. Ohsaki, "Rendering trees from precomputed z-buffer views," in *Proc. Eurographics Workshop on Rendering*, pp. 45-54, Dublin, Ireland, June, 1995.

[115] L. McMillan and G. Bishop, "Head-tracked stereoscopic display using image warping," in *Proc. International Symposium on Electronic Imaging: Stereoscopic Displays and Virtual Reality Systems II vol. SPIE 2409*, pp. 21-30, San Jose, CA, USA, February 7-8, 1995.

[116] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *Proc. Siggraph Annual Conference*, pp. 39-46, Los Angeles, CA, USA, August 9-11, 1995.

[117] G. Miller, E. Hoffert, S. E. Chen, E. Patterson, D. Blackketter, S. Rubin, S. A. Applin, D. Yim, and J. Hannan, "The virtual museum: Interactive 3D navigation of a multimedia database," *Journal of Visualization and Computer Animation*, vol. 3, no. 3, pp. 183-197, July-September 1992.

[118] B. O. Mok, M. Chen, J. Dorsey, and F. Durand, "Image-based modeling and photo editing," in *Proc. Siggraph Annual Conference*, pp. 433-442, Los Angeles, CA, USA, August 12-17, 2001.

[119] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta, "Occlusion detectable stereo - occlusion patterns in camera matrix," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 371-378, San Francisco, CA, USA, June 18-20, 1996.

[120] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Transactions on Communications*, vol. 36, no. 8, pp. 957-971, August 1988.

[121] S. K. Nayar, M. Watanabe, and M. Noguchi, "Real-time focus range sensor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1186-1198, December 1996.

[122] T. Nishita, Y. Dobashi, and E. Nakamae, "Display of clouds taking into account multiple anisotropic scattering and sky light," in *Proc. Siggraph Annual Conference*, pp. 379-386, New Orleans, LA, USA, August 4-9, 1996.

[123] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 139-154, March 1985.

[124] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353-363, April 1993.

[125] M. M. Oliveira and G. Bishop, "Image-based objects," in *Proc. Symposium on Interactive 3D Graphics*, pp. 191-198, Atlanta, GA, USA, April 26-28, 1999.

[126] M. M. Oliveira, G. Bishop, and D. McAllister, "Relief texture mapping," in *Proc. Siggraph Annual Conference*, pp. 359-368, New Orleans, LA, USA, July 23-28, 2000.

[127] S. Peleg and J. Herman, "Panoramic mosaics by manifold projection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 338-343, San Juan, PR, USA, June 17-19, 1997.

[128] B.-T. Phong, "Illumination for computer generated pictures," *Communications of*

the ACM, vol. 18, no. 6, pp. 311-317, June 1975.

[129] T. Porter and T. Duff, "Compositing digital images," in *Proc. Siggraph Annual Conference*, pp. 253-259, Minneapolis, MN, USA, July 23-27, 1984.

[130] M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images," *Computer Vision, Graphics, and Image Processing*, vol. 40, no. 1, pp. 1-29, October 1987.

[131] M. Potmesil and I. Chakravarty, "Lens and aperture camera model for synthetic image generation," in *Proc. Siggraph Annual Conference*, pp. 297-305, Dallas, TX, USA, August 3-7, 1981.

[132] K. Pulli, M. F. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle, "View-based rendering: Visualizing real objects from scanned range and color data," in *Proc. Eurographics Workshop on Rendering*, pp. 23-34, St. Etienne, France, June, 1997.

[133] P. Rademacher and G. Bishop, "Multiple-center-of-projection images," in *Proc. Siggraph Annual Conference*, pp. 199-206, Orlando, FL, USA, July 19-24, 1998.

[134] M. M. Rafferty, D. G. Aliaga, and A. A. Lastra, "3D image warping in architectural walkthroughs," in *Proc. Virtual Reality Annual International Symposium*, pp. 228-233, Atlanta, GA, USA, March 14-18, 1998.

[135] M. M. Rafferty, D. G. Aliaga, V. Popescu, and A. A. Lastra, "Images for accelerating architectural walkthroughs," *IEEE Computer Graphics and Applications*, vol. 18, no. 6, pp. 38-45, November-December 1998.

[136] R. Ramamoorthi and J. Arvo, "Creating generative models from range images," in *Proc. Siggraph Annual Conference*, pp. 195-204, Los Angeles, CA, USA, August 8-13, 1999.

[137] P. Rander, P. J. Narayanan, and T. Kanade, "Virtualized reality: Constructing time-varying virtual worlds from real world events," in *Proc. IEEE Visualization*, pp. 277-283, Phoenix, AZ, USA, October, 1997.

[138] T. Reed and B. Wyvill, "Visual simulation of lightning," in *Proc. Siggraph Annual Conference*, pp. 359-364, Orlando, FL, USA, July 24-29, 1994.

[139] M. Regan and R. Pose, "Priority rendering with a virtual reality address recalculation pipeline," in *Proc. Siggraph Annual Conference*, pp. 155-162,

Orlando, FL, USA, July 24-29, 1994.

[140] D. A. Rowland and D. I. Perrett, "Manipulating facial appearance through shape and color," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, pp. 70-76, September 1995.

[141] D. Ruprecht and H. Mueller, "Image warping with scattered data interpolation," *IEEE Computer Graphics and Applications*, vol. 15, no. 2, pp. 37-43, March 1995.

[142] H. Saito and T. Kanade, "Shape reconstruction in projective grid space from large number of images," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition vol. 2*, pp. 49-54, Fort Collins, CO, USA, June 23-25, 1999.

[143] Sara, "Confidently stable monotonic matching for standard stereo pairs," http://cmp.felk.cvut.cz/cmp/demos/Stereo/New/Matching/smm-standard.html, 2002. (last accessed June 1, 2003)

[144] R. Sara, "Finding the largest unambiguous component of stereo matching," in *Proc. European Conference on Computer Vision vol. 2*, pp. 900-914, Copenhagen, Denmark, May 27-June 2, 2002.

[145] K. Satoh, I. Kitahara, and Y. Ohta, "3D image display with motion parallax by camera matrix stereo," in *Proc. International Conference on Multimedia Computing and Systems*, pp. 349-357, Hiroshima, Japan, June 17-23, 1996.

[146] K. Satoh and Y. Ohta, "Occlusion detectable stereo using a camera matrix," in *Proc. Asian Conference on Computer Vision*, pp. 331-335, Singapore, December 5-8, 1995.

[147] K. Satoh and Y. Ohta, "Occlusion detectable stereo - systematic comparison of detection algorithms," in *Proc. International Conference on Pattern Recognition vol. 1*, pp. 280-286, Los Alamitos, CA, USA, August 25-29, 1996.

[148] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7-42, April-June 2002.

[149] G. Schaufler, "Exploiting frame-to-frame coherence in a virtual reality system," in *Proc. Virtual Reality Annual International Symposium*, pp. 95-102, Santa Clara, CA, USA, March 30-April 3, 1996.

[150] G. Schaufler, "Image-based object representation by layered impostors," in *Proc. Symposium on Virtual Reality Software and Technology*, pp. 99-104, Taipei, Taiwan, November, 1998.

[151] G. Schaufler, "Per-object image warping with layered impostors," in *Proc. Eurographics Workshop on Rendering*, pp. 145-156, Vienna, Austria, June 29-July 1, 1998.

[152] G. Schaufler and M. Priglinger, "Efficient displacement mapping by image warping," in *Proc. Eurographics Workshop on Rendering*, pp. 175-186, Granada, Spain, June 21-23, 1999.

[153] G. Schaufler and W. Sturzlinger, "Three dimensional image cache for virtual reality," in *Proc. Eurographics Annual Conference*, pp. 227-235, Poitiers, France, August 26-30, 1996.

[154] H. Schirmacher, W. Heidrich, and H.-P. Seidel, "High-quality interactive lumigraph rendering through warping," in *Proc. Graphics Interface*, pp. 87-94, Montréal, Québec, Canada, May 15-17, 2000.

[155] H. Schirmacher, M. Li, and H.-P. Seidel, "On-the-fly processing of generalized lumigraphs," in *Proc. Eurographics Annual Conference*, pp. C165-C173, Manchester, United Kingdom, September 4-7, 2001.

[156] H. Schirmacher, C. Vogelgsang, H.-P. Seidel, and G. Greiner, "Efficient free form light field rendering," in *Proc. Vision, Modelling, and Visualization*, Stuttgart, Germany, November 21-23, 2001.

[157] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, "2-D shape blending: An intrinsic solution to the vertex path problem," in *Proc. Siggraph Annual Conference*, pp. 15-18, Anaheim, CA, USA, August 1-6, 1993.

[158] T. W. Sederberg and E. Greenwood, "A physically based approach to 2-D shape blending," in *Proc. Siggraph Annual Conference*, pp. 25-34, Chicago, IL, USA, July 26-31, 1992.

[159] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli, "Fast shadows and lighting effects using texture mapping," in *Proc. Siggraph Annual Conference*, pp. 249-252, Chicago, IL, USA, July 26-31, 1992.

[160] S. M. Seitz and C. R. Dyer, "Complete scene structure from four point

correspondences," in *Proc. International Conference on Computer Vision*, pp. 330-337, Cambridge, MA, USA, June, 1995.

[161] S. M. Seitz and C. R. Dyer, "View morphing," in *Proc. Siggraph Annual Conference*, pp. 21-30, New Orleans, LA, USA, August 4-9, 1996.

[162] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1067-1073, San Juan, PR, USA, June 17-19, 1997.

[163] S. Seitz and K. N. Kutulakos, "Plenoptic image editing," in *Proc. International Conference on Computer Vision*, pp. 17-24, Bombay, India, January 4-7, 1998.

[164] J. Shade, S. J. Gortler, L.-W. He, and R. Szeliski, "Layered depth images," in *Proc. Siggraph Annual Conference*, pp. 231-242, Orlando, FL, USA, July 19-24, 1998.

[165] J. Shade, D. Lischinski, D. H. Salesin, T. DeRose, and J. Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments," in *Proc. Siggraph Annual Conference*, pp. 75-82, New Orleans, LA, USA, August 4-9, 1996.

[166] H.-Y. Shum and L.-W. He, "Rendering with concentric mosaics," in *Proc. Siggraph Annual Conference*, pp. 299-306, Los Angeles, CA, USA, August 8-13, 1999.

[167] H.-Y. Shum and S.-B. Kang, "Review of image-based rendering techniques," in *Proc. Visual Communications and Image Processing*, pp. 2-13, Perth, Australia, June 20-23, 2000.

[168] F. Sillion, G. Drettakis, and B. Bodelet, "Efficient impostor manipulation for real-time visualization of urban scenery," in *Proc. Eurographics Annual Conference*, pp. C207-C218, Budapest, Hungary, September 4-8, 1997.

[169] G. Slabaugh, B. W. Culbertson, T. Malzbender, and R. Schafer, "A survey of methods for volumetric scene reconstruction from photographs," in *Proc. International Workshop on Volume Graphics*, Stony Brook, New York, USA, June 21-22, 2001.

[170] G. Slabaugh, R. Schafer, and M. Hans, "Image-based photo hulls," in *Proc. International Symposium on 3D Processing, Visualization, and Transmission*,

Padova, Italy, June 19-21, 2002.

[171] P. P. Sloan, M. F. Cohen, and S. J. Gortler, "Time critical lumigraph rendering," in *Proc. Symposium on Interactive 3D Graphics*, pp. 17-23, Providence, RI, USA, April 27-30, 1997.

[172] D. B. Smythe, *A two-pass mesh warping algorithm for object transformation and image interpolation*, ILM Technical Memo 1030, Computer Graphics Department, Lucasfilm Ltd, San Rafael, CA, USA, 1990.

[173] J. Stam and E. L. Fiume, "Turbulent wind fields for gaseous phenomena," in *Proc. Siggraph Annual Conference*, pp. 369-376, Anaheim, CA, USA, August 1-6, 1993.

[174] J. Stam and E. L. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," in *Proc. Siggraph Annual Conference*, pp. 129-135, Los Angeles, CA, USA, August 9-11, 1995.

[175] R. Szeliski, "Rapid octree construction from image sequences," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 58, no. 1, pp. 23-32, July 1993.

[176] R. Szeliski and R. Zabih, "An experimental comparison of stereo algorithms," in *Proc. Workshop on Vision Algorithms*, pp. 1-19, Kerkyra, Corfu, Greece, September 21-22, 1999.

[177] S. J. Teller and C. H. Sequin, "Visibility preprocessing for interactive walkthroughs," in *Proc. Siggraph Annual Conference*, pp. 61-69, Las Vegas, NV, USA, July 28-August 2, 1991.

[178] J. Torborg and J. T. Kajiya, "Talisman: Commodity realtime 3D graphics for the PC," in *Proc. Siggraph Annual Conference*, pp. 353-363, New Orleans, LA, USA, August 4-9, 1996.

[179] G. Tsang, S. Ghali, E. L. Fiume, and A. N. Venetsanopoulos, "A novel parameterization of the light field," in *Proc. Image and Multidimensional Digital Signal Processing Workshop*, pp. 319-322, Alpbach, Austria, July 12-16, 1998.

[180] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proc. Siggraph Annual Conference*, pp. 311-318, Orlando, FL, USA, July 24-29, 1994.

[181] T. Whitted, "Improved illumination model for shaded display," *Communications*

of the ACM, vol. 23, no. 6, pp. 343-349, June 1980.

[182] L. Williams, "Casting curved shadows on curved surfaces," in *Proc. Siggraph Annual Conference*, pp. 270-274, Atlanta, GA, USA, August 23-25, 1978.

[183] L. Williams, "Pyramidal parametrics," in *Proc. Siggraph Annual Conference*, pp. 1-11, Detroit, MI, England, July 25-29, 1983.

[184] G. Wolberg, *Digital image warping*, Los Alamitos, CA, USA: IEEE Computer Society Press, 1990.

[185] G. Wolberg, "Image morphing: A survey," *Visual Computer*, vol. 14, no. 8-9, pp. 360-372, 1998.

[186] D. Wood, D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle, "Surface light fields for 3D photography," in *Proc. Siggraph Annual Conference*, pp. 287-296, New Orleans, LA, USA, July 23-28, 2000.

[187] J. C. Yang, M. Everett, C. Buehler, and L. McMillan, "A real-time distributed light field camera," in *Proc. Eurographics Workshop on Rendering*, pp. 77-86, Pisa, Italy, June 26-28, 2002.

[188] Y. Yang, A. Yuille, and J. Lu, "Local, global, and multilevel stereo matching," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 274-279, New York City, NY, USA, June 15-17, 1993.

[189] Y. Yu, P. E. Debevec, J. Malik, and T. Hawkins, "Inverse global illumination: Recovering reflectance models of real scenes from photographs," in *Proc. Siggraph Annual Conference*, pp. 215-224, Los Angeles, CA, USA, August 8-13, 1999.

[190] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape from shading: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690-706, 1999.

[191] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, 2000.

[192] Z. Zhang, "Software: Microsoft easy camera calibration tool," http://research.microsoft.com/~zhang/Calib/, 2000. (last accessed June 1, 2003)

[193] Z. Zhang, L. Wang, B. Guo, and H.-Y. Shum, "Feature-based light field

morphing," in *Proc. Siggraph Annual Conference*, pp. 457-464, San Antonio, TX, USA, July 21-26, 2002.

[194] L. C. Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 675-684, July 2000.

Appendix A:
Reliability and Approximated Reliability

This appendix proves that the difference between the reliability and the approximate reliabilities of given match is no larger than the value of the discontinuity penalty.

Lemma: *Under reliability-based algorithm, assume that the minimum-cost path passes match <p,d>, the following inequalities hold:* $R'(p,d)-\lambda \le R(p,d) \le R'(p,d)+\lambda$.

As shown in Figure 5-3, without loss of generality, we assume the minimum-cost path is *ia*. First, we consider the situation that the disparity on the minimum-cost path changes at the end of the segment, such as what happened at pixel *y* in Figure 5-3, where the disparity is *c* at pixel *y* and is *k* at pixel *y*+1. The cost of path *ka* can then be calculated using $\mathbf{C}(ka)=\mathbf{S}[z,a]-\mathbf{S}[y,c]-\lambda$. We know that *ka* must be the minimum-cost path for segment [*y*+1,*z*]. This is because if there were another path *mn* with a lower cost, path *icmn* should have been the minimum-cost path for the whole scanline.

Obviously, when $d=k$, the minimum-cost path that does not use matches on *ec* has the smallest cost, which is $\mathbf{S}[y,d]+\mathbf{C}(ka)=\mathbf{S}[y,d]+\mathbf{S}[z,a]-\mathbf{S}[y,c]-\lambda$. Therefore, the reliability *R* of any match on *ec* is not smaller than $R'-\lambda$. On the other hand, even when $d\neq k$, we can still use path *iedka* as one of the paths that does not use matches on *ec*. The cost of *iedka* is $\mathbf{S}[y,d]+\lambda+\mathbf{C}(ka)=\mathbf{S}[y,d]+\mathbf{S}[z,a]-\mathbf{S}[y,c]$. Therefore, the reliability *R* of any match on *ec* is not larger than R'.

Now let us consider the situation that the disparity on the minimum-cost path does not change at the end of the segment, such as what happened at pixel *x* in Figure 5-3, where the disparities are *e* for both pixel *x* and *x*+1. The cost of the path *ea* can then be calculated using $\mathbf{C}(ea)=\mathbf{S}[z,a]-\mathbf{S}[x,e]$. Now we assume the minimum-cost path for segment [*x*+1,*z*] is *mn*. We have $\mathbf{C}(mn)\ge\mathbf{C}(ea)-\lambda$, since otherwise the minimum-cost path

for the whole scanline should have been *iemn*.

Obviously, when $f=m$, the minimum-cost path that does not use matches on *ge* will have the smallest cost, which is $\mathbf{S}[x,f]+\mathbf{C}(mn)>=\mathbf{S}[x,f]+\mathbf{C}(ea)-\lambda=\mathbf{S}[x,f]+\mathbf{S}[z,a]-\mathbf{S}[x,e]-\lambda$. Therefore, the reliability R of any match on *ge* is not smaller than $R'-\lambda$. On the other hand, we can always use path *igfea* as one of the paths that does not use matches on *ge*. The cost of *igfea* is $\mathbf{S}[x,f]+\lambda+\mathbf{C}(ea)=\mathbf{S}[x,f]+\lambda+\mathbf{S}[z,a]-\mathbf{S}[x,e]$. Therefore, the reliability R of any match on *ge* is not larger than $R'+\lambda$.

As a result, for both situations, we have $R'-\lambda \le R \le R'+\lambda$. ■

Appendix B:
Estimated Parameters for Camcorders

This appendix gives the parameters for the Sony TRV120 digital camcorders. The parameters estimated include the intrinsic parameters and the radial distortion parameters.

The five intrinsic parameters form the following matrix, which projects a 3D point to 2D image plane:

$$\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

The estimated values for these parameters are:

α	β	γ	u_0	v_0
796.58	812.23	–0.43766	321.26	233.78

Two parameters, κ_1 and κ_2, are used to model the radial distortion. Their estimated values are:

κ_1	κ_2
–0.26722	0.36963

Correspondingly, the distortion correction can be performed using the following equation:

$$r' = r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

where, r and r' are the radius of a pixel to the center before and after the correction, respectively. For a given pixel (x,y), its radius r is calculated using:

$$r = \sqrt{\left(\frac{x - u_0}{\alpha}\right)^2 + \left(\frac{y - v_0}{\beta}\right)^2}$$